M-ary Runlength Limited Coding and Signal Processing for Optical Data Storage

A Thesis
Presented to
The Academic Faculty

by

Jorge Estuardo Licona Núñez

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
March 2004

M-ary Runlength Limited Coding and Signal Processing for Optical Data Storage

Approved by:

Dr. Steven W. McLaughlin, Advisor

Dr. Mary Ann Ingram

Dr. John Barry

Dr. Robert Dickson

Dr. Vijay Madisetti

Date Approved: March 30, 2004

To my parents

Miguel and Conchis

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Steven W. McLaughlin, for his guidance during my time at Georgia Tech. His support, encouragement and advice helped me become better as a professional and as a person. His insight and knowledge were instrumental to the research presented in this work. In particular, I would like to thank him for making sure that I always had the financial support needed to fulfill my lifelong dream of obtaining the Ph.D. degree.

I would also like to thank Dr. John Barry, Dr. Vijay Madisetti and Dr. Ingram for serving in my qualifying exam, proposal and dissertation defense committees. Their comments and ideas have helped focus this work. Many thanks to Dr. Robert Dickson for taking the time to serve in my dissertation defense committee.

I thank Dr. Sayle, Dr. Hertling, Marilou Mycko, Suzette Willingham and Sherrie Cooper at the School of Electrical and Computer Engineering who were always willing to help when needed.

I am grateful to my parents for their support, encouragement and prayers. Their advice and love have helped me to complete my studies. They stood behind me at all times. I thank my brothers for always being there for me when I needed to discuss my problems and for encouraging me to keep going forward and overcome all difficulties.

Thanks are also in order to all my friends and colleagues at GCATT. To name them all is impossible, many have graduated before me and a few stay behind, but I appreciate the friendship of all of them and they have all made my time here more enjoyable. In particular, I would like to thank Renato and Badri who took the time to read this work and make sure that its presentation was clear and concise. Renato and Andrew were also of invaluable help at various stages in my research and without their help and insights must

of the results included would not exist. Thanks to Mai, for the many discussions on the optical data storage system model and the collaboration on the first paper I wrote based on this research.

Thanks to all my friends in Atlanta and many who have left for the times shared in relaxation forgetting the pressure and struggle of my Ph.D. research. Nytzia, thank you for your support and encouragement over these last stressful months.

Finally I would like to thank God for always being by my side and with Him nothing is impossible. To Him be all the glory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The compact disc (CD) and digital versatile disc (DVD) are two types of binary optical data storage systems that have become industry standards. Recent attempts to increase the capacity of those systems have explored the use of multilevel recording instead of binary recording. Systems that achieve an increase in capacity of about three times that of conventional CD have been proposed and are being considered for production. Marks in these systems are multilevel and fixed-length as opposed to binary and variable length in CD and DVD. Here, we study the modeling and implementation of multilevel optical data storage. The main objective of this work is to evaluate the performance of multilevel (M-ary) runlength-limited (RLL) coded sequences in optical data storage. This is accomplished by dividing the research into three major areas.

First, the waterfilling capacity of a multilevel optical recording channel (M-ary ORC) is derived and evaluated, providing insight into what kind of user bit densities are achievable as well as a theoretical limit against which simulated systems can be compared. The model is kept as general as possible. Therefore, the results apply to a variety of recording options.

The next step is to evaluate the performance of RLL codes on the M-ary ORC. To accomplish this, a more specific channel model is taken into account, one that includes the runlength constraint in the transmitted signal. The performance of these codes is evaluated and compared to the theoretical limits. We also compare the performance of specific RLL codes, namely M-ary permutation codes, to that of real systems using multilevel fixed-length marks for recording. The Viterbi detector is used to estimate the original recorded symbols from the readout signal.

We then include error correction in the system, while signaling with M-ary RLL codes.

Error correction is used to reduce the probability of symbol shifts in the readout signal, the leading cause of errors in RLL coded sequences. We use a combined ECC/RLL code for phrase encoding. We evaluate the use of trellis coded modulation (TCM) for amplitude encoding. The detection of the readout signal is also studied. In particular, a post-processing algorithm for the Viterbi detector is introduced. This algorithm ensures that the detected word satisfies the code constraints. Specifying the codes and detector for the M-ary ORC gives a complete system whose performance can be compared to that of the recently developed systems found in the literature and the theoretical limits calculated in this research.

CHAPTER 1

INTRODUCTION

Information storage and retrieval is an important segment of the communications industry. Research efforts have concentrated on providing an inexpensive, highly portable and high capacity method for data storage. Two main types of recording media have emerged as the most used methods for storage: magnetic and optical. Optical recording uses disks which are imprinted with information bits. An optical laser is then used to read the information stored in the disks. These disks are inexpensive to manufacture, portable, robust and until recently had larger capacities than portable magnetic media. To maintain their competitive advantage over magnetic media with their always increasing capacity, it is of interest to increase the capacity of optical disks. This is the motivation of this research, to evaluate the use of M-ary runlength-limited codes as a mean to increase the capacity of an optical disk. To understand how this can be achieved, it is first of interest to know how information is stored in the disks.

Traditionally, optical and magnetic recording channels have been of the saturation type, which limits the recorded information to binary values. Conventional optical recording systems like the compact disc (CD) and digital versatile disc (DVD) were developed under this constraint. They use binary signaling in conjunction with runlength-limited (RLL) codes. In a binary runlength-limited code each one in the sequence is followed by at least d and at most k zeros. The d constraint is imposed to reduce intersymbol interference in the system. In addition it can be used to provide an effective increase in the recording density. The k constraint is present to enable timing recovery directly from the recorded signal [21, 24].

The combination of binary signaling with runlength-limited codes can be essentially

seen as a pulse width modulation (PWM) scheme, where information is conveyed in the length of the marks on the disc and the spacing between them. The dominant source of errors in these systems is jitter, producing variations in the pulse widths of the recorded waveforms [25]. To correct errors in the system, error control codes (ECC) codes are used as an outer code. Then, the overall system is a serial concatenation of an outer ECC code with an inner RLL modulation code, as shown in Figure 1. In CD, the ECC code is a cross interleaved Reed Solomon code (CIRC), while in DVD it is a Reed Solomon product code (RS-PC). The inner RLL code is either Eight to Fourteen Modulation (EFM) for CD or EFM Plus for DVD.

Increasing storage density using binary signaling (without changes to the optical or mechanical parameters of the system) requires more information in the pulse widths, which is very difficult because of the inherent jitter limitations. However, if the optical and mechanical parameters of the system can be varied, then the storage density can be increased more easily. This is the approach taken in Blu-ray [12], a new technology that increases capacity to about five times that achievable by DVD recordings. This technology uses a blue laser and lenses of a higher numerical aperture for readout. This allows a tighter focus of the laser spot on the disk. Therefore, information can be stored on the disk with smaller marks and tighter track spacing. The main disadvantage of this approach is that new disk drives are required for readout, since the parameters of the laser and focusing lens are distinct from those of the industry standards.

Amplitude variations in the signal are not the major source of random noise in optical data storage systems. Therefore, one could envision encoding information in the amplitude of the signal, if the medium allows it. This is not possible with saturation-type channels, unless fixed-length marks are used. The saturation nature of the media only allows two states to exist. Therefore, only binary signals can be recorded. However, recent research has produced media capable of supporting multilevel data [20, 30, 42, 51].

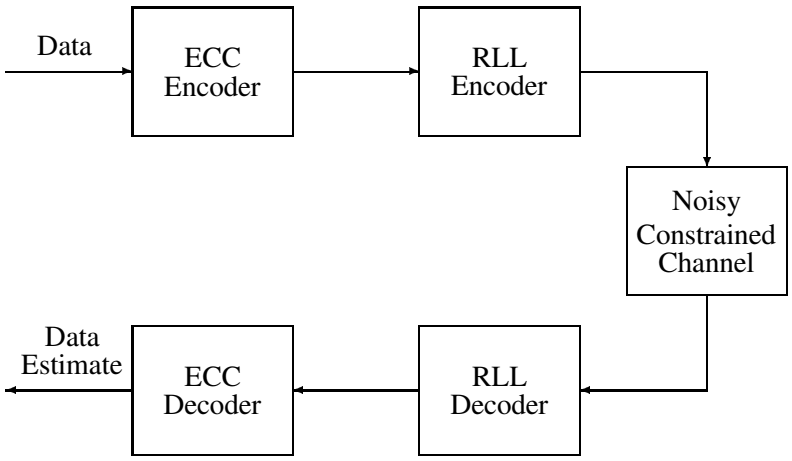


Figure 1: A model of a recording channel as a communication system.

These new media have led to research and implementations of systems storing non-binary (M-ary) signals using fixed-length, pulse amplitude modulation (PAM), in an attempt to store more information per unit area. Systems employing $M = 8$ and $M = 12$ levels and fixed-length marks on conventional and custom write-once and re-writable optical discs that achieve 2.5 bits/0.6 micron mark have been reported [32, 36, 37]. Typical CD systems achieve about a density of about 1.5 bits/0.6 micron mark. The media in these systems are compatible with CD-R and CD-RW. This means the recorded marks are still binary (saturation type recordings). However, the recording pattern in the mark reads out like a multilevel signal. This is accomplished by recording marks of different sizes and shapes within a fixed-length mark period. The write-once and re-writable optical discs are highly nonlinear. Nonlinear precompensation is used to linearize the channel so conventional coding and signal processing for a linear bandlimited channel can be employed [31]. These fixed-length mark systems are characterized by the fact that there is no inner modulation code. The outer ECC codes in these systems are not RS codes; instead, systems with trellis coded modulation (TCM) and Turbo codes have been proposed. We will refer to these type of systems as Multilevel (ML) CD-RW.

ML CD-RW provides about a threefold increase in the capacity of a CD. The large number of amplitude levels required to do this cause the eye opening between levels to close down and heavy coding is required to correct errors, see Figure 2. If less levels are used then the eye openings in the eye diagram will be larger, reducing the number of detection errors. However, the reduction in levels causes a reduction in the bit density. The question then is can these improvements be increased by encoding information in the length of the marks recorded, as well as, in the amplitude of those marks? Essentially, can a combination of the traditional optical recording technique (RLL codes) with multilevel signaling provide a further increase in the information density of the new system? The advantage of combining RLL codes with multilevel signaling is that fewer amplitude levels are required to achieve the same density increase, since information is also encoded in the length of the marks. A recording system that can accommodate multilevel RLL signals has already been introduced in the literature [51]. It differs from the previous optical data storage systems in that it does not use a saturation type media. Therefore, this research addresses the coding and signal processing issues with implementing a multilevel runlength-limited system for data recording. To do so requires the use of M-ary (d,k) constrained codes or (M,d,k) codes, which were first introduced in [44]. Most (M,d,k) codes do not provide any error-correction capabilities. In general, they act solely as modulation codes. Therefore, a concatenation of an ECC code and an (M,d,k) code is required to achieve good performance.

The concatenation of the ECC and RLL code can be done in more than one way. Figure 1 showed the traditional approach, an outer ECC code and an inner RLL code. In [1] the performance of reverse concatenation is evaluated for magnetic recording. Reverse concatenation consists of using an outer RLL code and an inner ECC code. The inner ECC code is systematic in order to maintain the constraints of the RLL code. This type of system requires a separate RLL encoder for the parity symbols. Another approach is to use one code that is both an RLL and an ECC code. The main advantage of this approach

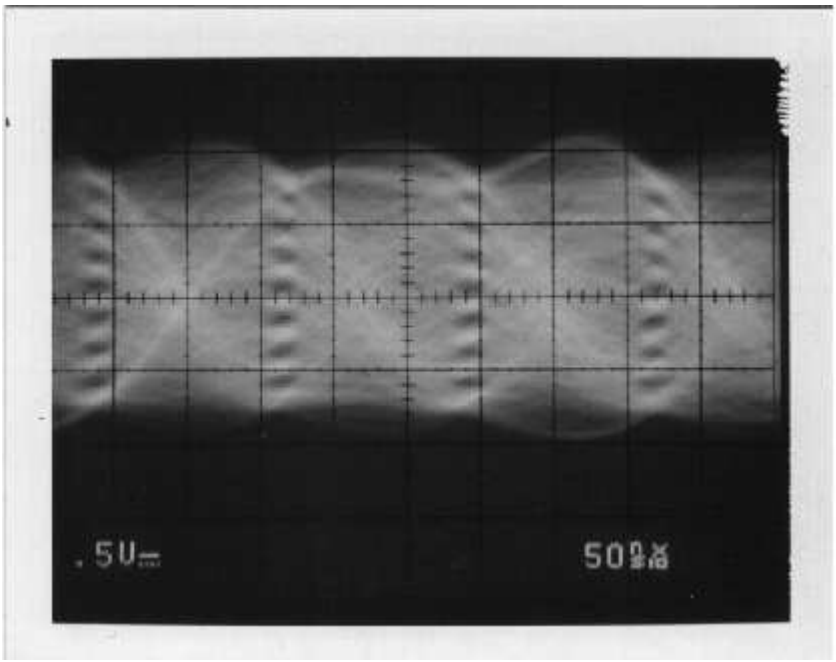


Figure 2: Eye diagram for ML CD-RW [54].

is that only one encoder/decoder is required. In this work we evaluate the traditional approach, although we only consider the performance of the RLL code. We then evaluate the performance of a code that combines RLL and ECC encoding.

So far, we have discussed the first of the two key aspects of an optical recording system, encoding the signal. Now we must address the issue of signal detection, how to recover the information signal stored on the disc. The first detection methods for binary RLL sequences employed a peak detection circuit. This circuit is the one originally used in CD players. It detects zero crossings in the derivative of the recorded signal to find the peaks in the recorded signal. At the point a peak is found, the circuit records a transition in a mark. After the peak detector, the RLL code (EFM) is demodulated and CIRC decoding is performed [2, 50]. The peak detector is simple and easy to implement. Its main drawback is performance degradation for high user bit densities.

The first implementations of M-ary optical recording systems use more sophisticated detection methods. In particular, a concatenation of a whitened matched filter with a zero

forcing linear equalizer (ZF-LE) is used in [32]. The whitened matched filter is used because it provides a set of sufficient statistics for signal detection [28]. The ZF-LE is used because it is a simple, computationally efficient algorithm that provides optimal performance for this particular case. It is well known that when the noise is white the performance of the ZF-LE degrades. However, for these M-ary optical recording systems the noise is colored by the channel. Therefore, there is no degradation in the performance of the ZF-LE. In fact, this is the one situation in which the performance of a ZF-LE is optimum. After the ZF-LE, the ECC code is decoded.

The detection of the concatenated ECC and (M,d,k) codes is addressed in this work. A whitened matched filter (WMF) front end is used to provide a sufficient-statistics discrete-time (SSDT) channel model. Detection is performed using the Viterbi algorithm [49]. The trellis used in the Viterbi algorithm is that of the intersymbol-interference (ISI) channel obtained from the WMF front end. The trellis can also incorporate some of the constraints of the (M,d,k) code. In particular, states that do not satisfy the d constraint can be eliminated. This reduces the complexity of the algorithm by removing states and transitions from the trellis. However, there is no simple procedure to incorporate the k constraint into the trellis. The Viterbi algorithm does not always output a codeword of the (M,d,k) code being used; instead it only ensures that the detected word satisfies the d constraint. For this reason, a post-processing algorithm needs to be implemented to ensure that the detected codeword belongs to the codebook of the (M,d,k) code being used and satisfies the k constraint. This work introduces a particular implementation of the post-processing algorithm.

This dissertation is organized as follows. Chapter 2 presents background information on recording on optical disks. An introduction to saturation-type recording is given. Mainly, a discussion on CD, DVD and Blu-ray is presented. We give an explanation on how an increase in capacity is achieved with each new standard. Then, an introduction into multilevel recording on optical disks is given. We analyze multilevel recording on saturation-type disks. The particular approach to encoding, decoding and detection used in real systems

is discussed. Multilevel recording on optical disks, especially the system in [51], is introduced. An introduction to (M,d,k) codes is also included in this chapter. It deals with the basics of (M,d,k) codes, introducing some of the different types of (M,d,k) encoding methods. The advantages and disadvantages of each method are presented. This analysis is used to select adequate (M,d,k) encoding methods for the combined ECC/RLL system. Finally, a brief discussion on detection methods used in optical disk recording channels is given. Once again, the advantages and disadvantages of each method are described.

Chapter 3 presents a model for the optical recording channel. We discuss specific model issues, such as the selection of the modulation transfer function for the channel frequency response. The noise model used is also introduced and the physical processes behind the noise model are discussed. A continuous-time channel model is introduced first. It provides the maximum attainable capacity, if a way to record continuous marks on optical discs can be found. Then we specialize to the discrete-time model. This model is a more accurate representation of the recording process on an optical disk. For both models channel capacity is calculated. Plots of SNR vs. capacity are shown for different values of the important system parameters.

Chapter 4 introduces the sufficient statistics discrete-time (SSDT) model for both fixed-length and variable-length recording. This models transform our channel models into simple finite-impulse response channels with additive white Gaussian noise. The FIR channel is then used to generate a trellis for Viterbi detection of the signals recorded on the channel. The variable-length signals are generated from an M-ary permutation code, whose parameters are included in the detection trellis. The performance of the uncoded system and an M-ary RLL coded system are presented. This performance is compared to the achievable capacity, from chapter 3.

In Chapter 5, a complete system using permutation codes is presented. We discuss the disadvantage of combining permutation codes with Viterbi detection. Two alternative approaches for detection are discussed. Both are discarded as viable alternatives due to their

complexity. However, their discussion is important since they motivate the final approach used for detection in this work. This chapter also introduces a complete system using combined error-correcting codes (ECC) and RLL codes. Two code construction techniques for combined ECC/RLL codes are presented. We discuss the advantages and disadvantages of each method, and select the one that we believe will provide the best performance. The performance of the system is evaluated. It is observed that for high user bit densities the detection with the Viterbi detector needs to be modified to improve performance. Finally, a post-processing algorithm for the Viterbi detector is introduced. This algorithm guarantees that the output of the detector is a valid codeword in our combined ECC/RLL code. We show how system performance is dramatically improved by the use of the post-processing algorithm. A categorization of the type of detection errors in the system is performed. An evaluation of this analysis leads us to believe that ECC encoding on the amplitude signal can further decrease the error rate.

Chapter 6 introduces amplitude encoding. We use trellis coded modulation (TCM) as our error-correcting code. We implement two TCM codes for $M = 5$. Both codes use convolutional encoders to select the encoded symbols from the M-ary alphabet. The first code encodes two amplitude symbols at a time, the second code four amplitude symbols at a time. A complete description of the code details is given. We specify how we obtain soft-information for the amplitudes from the noisy readout sequence. We discuss the TCM decoder, which is a modified version of the Viterbi detector normally used for TCM decoding. The modification is necessary because our soft-information is based on the channel symbols and not on the alphabet used to design our code. Performance is evaluated for two user bit densities. A discussion of the results provides insight into when the use of amplitude encoding provides a performance gain. Finally, Chapter 7 is a summary of the work presented in this document. A brief description of the contributions presented in this work is given. Also, recommendations for future research are included.

CHAPTER 2

BACKGROUND

This chapter introduces the different methods used for recording on optical discs. Information on important system parameters is provided. In particular, a description on how those parameters affect disk capacity is given. This discussion encompasses binary recording, multilevel recording on binary media and pure multilevel systems. Binary systems have been the traditional approach in optical recording. The information in these systems is stored in the length of the marks written on the disk. Therefore, marks written on the disk are of varying length. Their main limitation, the number of levels used to record marks, is the main motivation for this work. We give a brief introduction to M-ary recording on binary optical discs. This is an alternative approach to increasing capacity. We introduce it in this work, since we compare our performance simulations to real systems that employ this approach. Therefore, it is important that the reader know how the physical marks on these systems would compare to those on systems employing the coding approach introduced here. Finally, M-ary recording in optical disks is discussed. This technology is necessary for the implementation of the ideas presented in the following chapters. Therefore, we believe it is necessary to have an understanding of how M-ary recording can be physically accomplished.

We also introduce M-ary RLL coding techniques. Binary RLL signals are widely used in the recording industry, and aid in the removal of intersymbol interference (ISI) and in timing recovery. The use of an M-ary alphabet increases the capacity of the RLL constraint. Therefore, an understanding of the methods for code construction is required. Knowledge of the advantages and disadvantages of each method is a must, as it will enable us to chose the encoding method that provides the best results in our application.

Finally, a brief introduction to the techniques used for symbol detection is given. We start with peak detection, which was the original bit-by-bit detection method used in CD. We then introduce some sequence detection methods that are used in more modern systems. This discussion gives the most important characteristics of the methods that we will use in our designed system.

2.1 Binary Optical Recording

Traditional optical recording, such as that employed in early CD-ROMs, employs a system of pits and lands to write information on the disc. In this system, the pits and lands representing the marks are stamped onto the disc and covered with a reflective film and a protective layer. When the laser beam is focused on the pit, light is scattered from its edges, reducing the reflectivity and the strength of the signal received at the photo-detector, as seen in Figure 3. When the laser is focused on the lands, the reflectivity is high because of the reflective covering, causing a high strength signal at the photo-detector. In this way, pits and lands can be differentiated with ease and the marks can be read out of the disc. Figure 4 shows a view from above the pits in a CD-ROM. It shows the measurements of marks in the system, particularly mark length, width, and track-to-track spacing [2, 50]. The length of a mark determines how many zeros follow a one in the user signal. Therefore, information is encoded into the mark length. In the readout process we should be able to accurately detect the mark lengths in order to decode the user information. Therefore, transition jitter or random displacements of the edges of a mark from their ideal position are the dominant source of noise in these systems.

The development of CD-R and CD-RW has changed the way information is written in discs. Disks are not physically stamped with pits and lands. Instead, the state of the disc media changes to one of two allowable states. This change can be of two types: polarization changes in magneto-optical (MO) discs or phase changes in purely optical

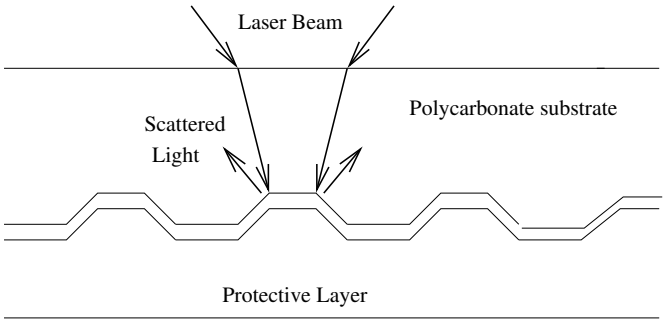


Figure 3: Readout of a pit in a CD-ROM.

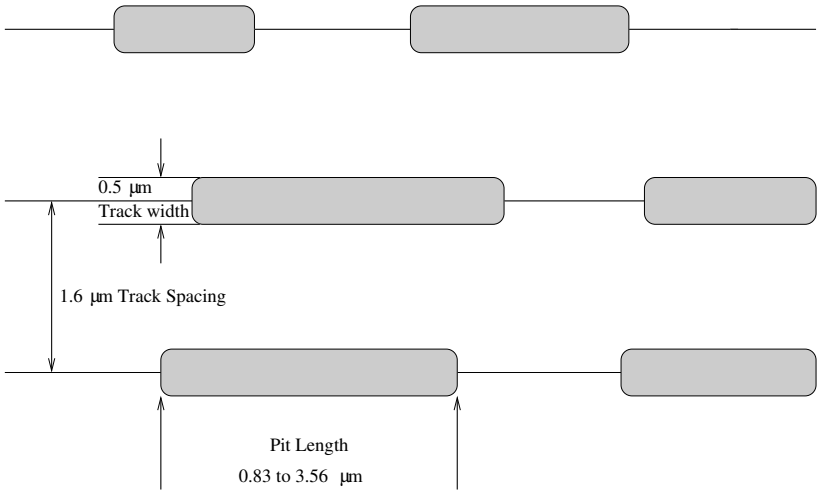


Figure 4: CD-ROM pit sizes and track spacing.

discs. Here we concentrate on phase change media, since it is the technology behind CD-RW and multilevel (ML) CD-RW.

Phase change media exist in two states of varying reflectivity. The amorphous state has a low reflectivity, while the crystalline state is one of high reflectivity. The interchanging of states can then be used in a manner analogous to the pit and land system, i.e., the amorphous state represents a pit and the crystalline state a land. The readout and write processes require a highly focused laser beam to illuminate the phase change media. The power of the laser beam will vary, depending on the operation being performed. A low-power laser beam is used for readout, to avoid heating the medium and causing a change in phase. For writing, a higher power laser is used to perform the appropriate phase change in the medium [8].

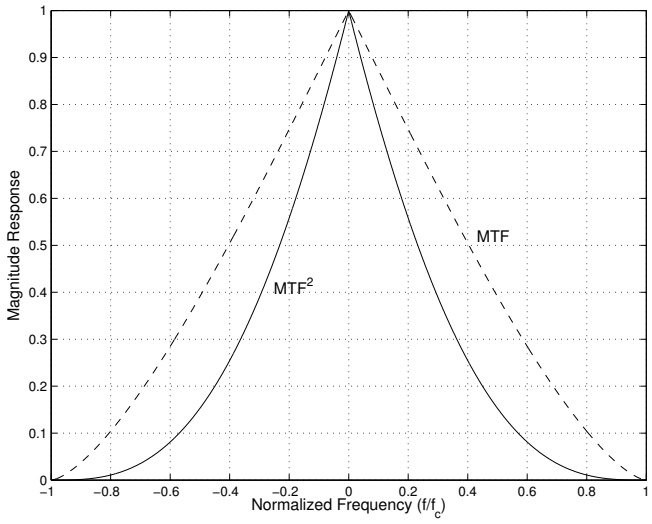


Figure 5: MTF and MTF^2 for the optical recording channel.

It is of interest to make the marks (pits and lands) in the system as small as possible, to pack more bits into the recordable surface of the optical disc. Two parameters of the optical system directly affect the size of the mark: the laser wavelength λ and the numerical aperture of the focusing lens *NA*. These parameters affect the size of the focused laser spot. The smallest size that can be achieved is the "full width at half maximum density" (FWHM), which is given by $FWHM \cong 0.6\lambda/NA$. A smaller focused laser spot means that we can detect smaller marks in the disk. When this happens, we can decrease the length of marks on the disk to increase capacity. They also determine the critical frequency f_c of the frequency response of the readout channel. The critical frequency $f_c = 2NA/\lambda$ is the frequency above which the channel frequency response is zero. One possible equation for the frequency response of the optical recording channel is [29]

$$H_c(f) = \begin{cases} \frac{2}{\pi}\left(\cos^{-1}|f| - |f|\sqrt{1-f^2}\right) & |f| \le f_c \\ 0 & otherwise \end{cases} \quad (1)$$

we call $H_c(f)$ the modulation transfer function (MTF). Figure 5 shows a plot of both the MTF and the MTF^2. We will use the MTF^2, as our channel frequency response, in our system model in Chapter 3 as it closely matches the frequency response of real multilevel

Table 1: Parameters for CD, DVD and Blu-Ray.

Technology	λ	NA	l_{MF}	$Capacity$	RLL Code	ECC Code
CD	780 nm	0.45	390 nm	700 MB	EFM	CIRC
DVD	680 nm	0.55	310 nm	4.7 GB	EFM Plus	RS-PC
Blu-ray	405 nm	0.85	120 nm	23, 25, 27 GB	17PP	Picket Code

optical recording systems. With this channel frequency response, the smallest resolvable mark size, which we will call the minimum feature (MF), has a length $l_{MF} = \lambda/4NA$ [29].

The underlying physics of recording in CD, DVD and Blu-Ray are the same. The main difference between the technologies are the values of the two parameters determining the FWHM. Table 1 gives the parameters for each of the three technologies. The laser wavelengths for CD and DVD correspond to lasers in the red color range. The wavelength for Blu-Ray is in the blue color range, hence the name for the technology. The reduction in the size of the FWHM is the main reason behind the increase in capacity from one technology to the next. This allows the use of smaller marks to record information on the disk. Smaller marks in turn allow us to fit more marks in a disk of the same size, thus increasing the disk capacity. Changes in coding and signal processing also provide some increase in capacity from one technology to the next. In particular, more efficient RLL and ECC codes provide an increase in capacity. However, most of the capacity increase comes from the changes in λ and NA.

2.1.1 Multilevel Recording on Binary Media

Multilevel recording media, where the marks on the disk are not binary, but consist of different levels have been presented in [20, 30, 42, 51]. Another approach to multilevel recording is to record "multilevel" signals on binary media [32,35–37]. The physical marks on the disk are binary. However, their size or location on the disk is different for each type of mark so that the signal reads out as a multilevel signal. Each level is represented by a different type of binary mark.

The approach in [35] is to write binary RLL signals and read them as multilevel signals.

Each one of the M different pit lengths specifies a different symbol in the M level alphabet. Then, a recorded signal with four different pit lengths provides an M-ary alphabet with $M = 4$. The number of levels can be increased if the pit period is shortened and a partial response maximum likelihood (PRML) channel is used to combine two pits. The PRML channel is physically realized by a laser that reads two pit edges at once. With this method up to seven levels can be accomplished. Using more levels degrades the SNR in data detection. This problem is compounded by the non-linearity introduced by the readout process and by pattern dependence. Detection requires the use of a specialized Viterbi detector that includes the channel non-linearity. This detector has two reference matrices used for transition detection. Two matrices are required since the readout is different when the laser focuses on a pit than when it focuses on a land.

Another approach is the one used in [32, 36, 37]. In these systems, the multilevel signal is obtained by writing binary marks of different sizes and positions within a fixed-sized data cell. The size of the data cell must be larger than that of a minimum feature. Typical sizes are about one and a half times a minimum feature. This increased data cell size acts to reduce ISI. At readout, each different mark is identified as a different level in a multilevel signal. Figure 6 shows the process of writing multilevel signals on binary media, comparing it to traditional binary recording. The real marks are binary. However, since there are M different mark types, the readout signal interprets them as M different levels in an M-ary alphabet. This is accomplished by varying the percentage of the laser power reflected by each mark. Then, a low mark in the alphabet would reflect only a small fraction of the readout laser power. A high mark in the alphabet would be shaped and sized in such a way that a large percentage of the readout laser power is reflected.

A block diagram of the process of writing multilevel marks on binary media can be seen in Figure 7. Write precompensation is necessary to reduce nonlinearities in the readout process. This precompensation is performed in the second block in the diagram, the write strategy lookup. For precompensation, the system needs to establish contours of equal

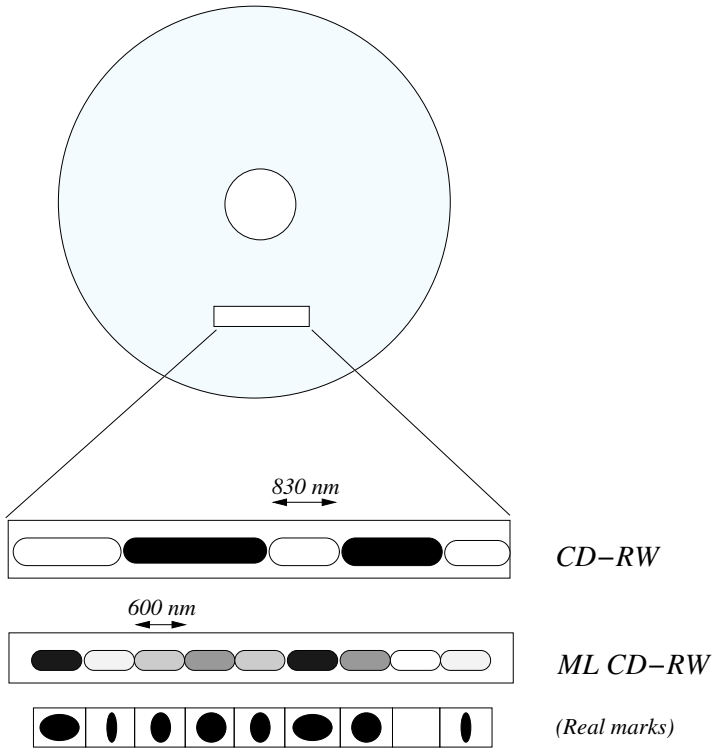


Figure 6: Comparison of marks in CD-R/RW and ML CD-R/RW.

reflectivity for written marks. These are based on the power of the laser and the times it is on. Then, reflectivity sections of high linearity are selected and the corresponding power levels and times are recorded. Write precompensation is used to achieve linearity in the readout signal. As an example, a mark '1' is not always written with the same power. The mark '1' in '8 1 8' requires less power than the one in a '3 1 3' sequence since the readout detector will obtain additional reflectivity from the adjacent '8' marks in the first case. Therefore, the precompensation block would ensure that the appropriate power level and pulse duration is used for each mark, taking its two neighbors into consideration.

2.2 Multilevel Optical Recording Media

Systems allowing purely multilevel signaling on optical disks were presented in [20, 30, 51]. In these systems the written marks on the disk are multilevel. Multilevel recording is achieved by recording marks with more than two reflectivity levels. In [30], this is achieved

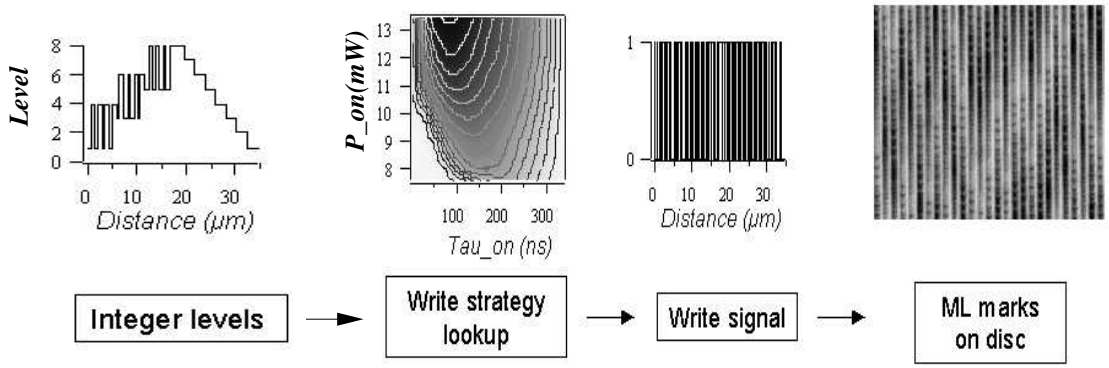


Figure 7: Writing steps for ML CD-R/RW.

by varying the depth of the pits in which information is stored. This technique has the disadvantage that it does not allow simple re-writable systems to be implemented. Once the pit depth is imprinted in the disc it is pretty much impossible to change it. Another way to achieve different reflectivity levels is to use phase-change optical discs. In these materials an amorphous state is one of low reflectivity and the crystalline state has a high reflectivity. A partially crystallized state would then provide another level of reflectivity. This approach is the one taken in [51]. This particular system is of interest since it allows multilevel recording of signals with varying mark length. In essence, it allows multilevel recording of RLL signals, which we will refer to as ML-RLL. The system described allows the recording of $M = 3$ levels.

The system uses an optical head with an $NA = 0.5$ and a laser wavelength of 675 nm. Two different methods of recording were tested. Recording partially crystallized marks in an amorphous thin film performs better than recording fully crystallized marks in an amorphous film. The system employs a single write laser power. This means the laser is switched between a low bias (read) power and a high (write) power. The reflectance level and length of the written marks are varied by changing the width, duty cycle and number of short sub-pulses used to write each individual mark. These parameters dictate the level of crystallization of each mark and therefore the reflectivity of that particular mark.

Until now, only systems with $M = 3$ levels of ML-RLL recording have been presented.

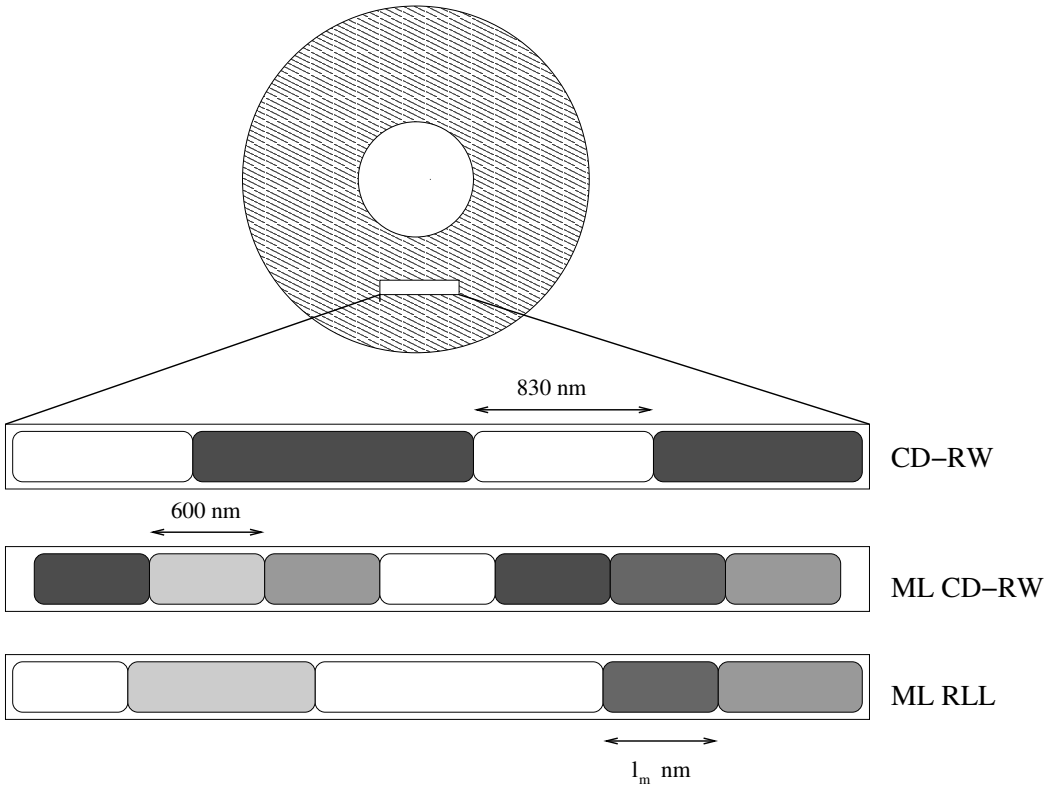


Figure 8: Comparison of marks in CD-R/RW, ML CD-R/RW and ML-RLL.

However, one can envision that in the future the technology will advance so that larger values of M can be accommodated. Such a recording media would be necessary to implement the systems proposed in this dissertation. An example of a signal written in a 4L-RLL format, as well as in CD and CD-RW formats, can be seen in Figure 8. The length of the minimum mark in the 4L-RLL sequence has been labeled as l_m, since no actual standard exists yet. A smaller mark length allows for a higher user density. However, the mark must be larger than the minimum feature. Also, in general smaller marks increase ISI, degrading system performance.

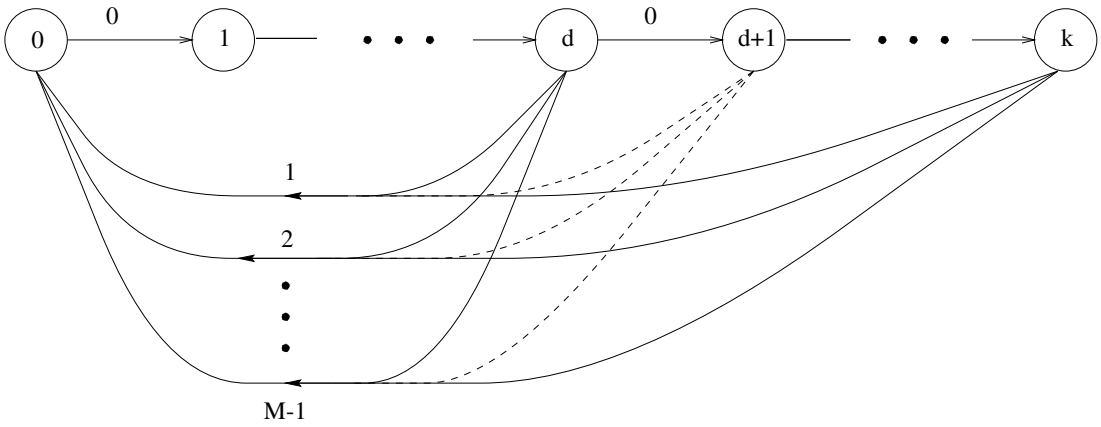
2.3 Constrained Sequences

A sequence of 0's and 1's which is not random, but instead satisfies a particular set of rules on the placement of symbols within the sequence is called a constrained sequence. We

are particularly interested in (d,k) constrained sequences. A binary (d,k) sequence is one where each one in the sequence is followed by at least d zeros and at most k zeros. The d constraint is used to avoid intersymbol interference, while at the same time increasing the recording density. The k constraint is used to improve timing in the system, since in most recording systems timing is extracted from the recorded signal. Therefore, a sequence with long runlengths of the same symbol would affect timing negatively. In general, a (d,k) constrained sequence is precoded using a mod 2 NRZI precoder. Such a precoder satisfies the equation $y_j = y_{j-1} + x_j \pmod 2$, where y_i is a symbol from the precoded sequence and x_i is a symbol from the constrained sequence. The output sequence of the precoder is called a runlength-limited sequence. This runlength-limited sequence is characterized by the fact that there are at least $d+1$ and at most $k+1$ consecutive valued symbols in the sequence. This encoding process can be performed with non-binary constrained sequences to generate M-ary runlength-limited sequences. To do so, we first give a description of non-binary constrained sequences.

2.3.1 M-ary $(\mathbf{D},\mathbf{K})$ Constrained Sequences

An M-ary $(\mathbf{D},\mathbf{K})$ code $(M,\mathbf{D},\mathbf{K})$ is a runlength limited code with symbols coming from the alphabet $\mathscr{M} = \{0,1,\ldots,M-1\}$ that satisfies a minimum runlength constraint given by the $(M-1)\times(M-1)$ matrix $\mathbf{D}$ and a maximum runlength constraint given by the $(M-1)\times(M-1)$ matrix $\mathbf{K}$. The elements d_{ij} of $\mathbf{D}$ specify the minimum number of zeros in between the i and j symbols in a sequence satisfying the constraints, where $i,j = \{1,2,\ldots,M-1\}$. Similarly the elements of $\mathbf{K}$ specify the maximum number of zeros between two non-zero symbols. This generalized definition of the M-ary $(\mathbf{D},\mathbf{K})$ code was introduced in [18]. A special case of these codes, the (M,d,k) code, in which $d_{ij} = d$ for all i,j and $k_{ij} = k$ for all i,j, was described in [44]. This type of code is the one regularly appearing in the literature and the one we use in this work. However, this does not mean that $(M,\mathbf{D},\mathbf{K})$ codes have no practical applications; as a matter of fact [16, 18, 43] all deal



FSTD G

Figure 9: Finite-state transition diagram (FSTD) of (M,d,k) code

with implementations of $(M,\mathbf{D},\mathbf{K})$ codes.

The purpose of the d constraint in an (M,d,k) code is to combat intersymbol interference and to increase the channel symbol rate, just like in binary constrained sequences. The k constraint is placed in order to enable timing recovery from the stored signal. An encoder is used to convert the information sequence into a constrained sequence. A fixed rate encoder converts p information bits $\mathbf{b} = (b_0,\ldots,b_{p-1})$ into q coded symbols $\mathbf{x} = (x_0,\ldots,x_{q-1})$ satisfying the code constraints. The signal can then be transmitted or it can be precoded using a mod-M NRZI style precoder [34] where $y_j = y_{j-1} + x_j (\mathrm{mod}\, M)$. Precoding turns the (M,d,k) sequence into an RLL sequence satisfying $(d+1,k+1)$ constraints with M levels. This is analogous to the binary case described above.

An allowable sequence in an (M,d,k) code is made up of phrases, which are a concatenation of zeros and one nonzero symbol. There are two possible ways to make up a phrase. In the first approach, each phrase begins with at least d and at most k zeroes and ends with a single nonzero symbol. The second approach places the nonzero symbol at the beginning of the phrase, followed by at least d and at most k zeros. For example, 0002 001 00006 005 satisfies a $(7,2,4)$ constraint based on the first definition. Notice that these definitions do not take into account the precoding.

The constraints in these codes can be represented by the use of a finite-state transition diagram FSTD. A discussion of FSTD's helps to understand the implementation of constrained sequences. At the same time the underlying mathematical theory can be used to calculate the capacity of the constraint. The capacity of a constraint is an upper limit on the rate that a constrained code can achieve. Therefore, it is important to understand how to calculate the capacity of a particular constraint. Figure 9 shows the FSTD of an (M,d,k) code. The FSTD G is a directed graph with a finite number of states, edges and edge labels drawn from the alphabet $\mathscr{M}$. The set of states of G will be denoted $V(G)$ and the set of edges is $E(G)$. Any allowable sequence in the RLL constrained system can be obtained from G by simply reading off labels through a path in G. The set of all finite sequences generated by G is the constrained system S. The FSTD representation of a constrained system S is not unique. An FSTD is called deterministic if at each state the outgoing edges are labeled distinctly. If there is always a path in G from any starting state i to any state j then the FSTD is irreducible. An FSTD has memory m and anticipation a if given $\mathbf{x} = x_{-m},\ldots,x_0,\ldots,x_a$ in S the set of paths $\mathbf{e} = e_{-m},\ldots,e_0,\ldots,e_a$ that generate $\mathbf{x}$ all agree on e_0. The anticipation defines the look-ahead of the FSTD and the memory defines the look-back, specifically the knowledge of how many previous or future transitions in the FSTD are needed to know what edge at the current state produces the specified output sequence. A deterministic FSTD has $a = 0$. The FSTD G^q is the q-th power of G. It has the same states as G, but the labels on the edges are q-length blocks that represent a path of length q on G from state i to state j. The constrained system generated by G^q is S^q and its alphabet is the set of q-blocks on S. The Shannon cover G_s is the minimal deterministic representation of S, i.e., the one with the smallest number of states. If S can be represented by an irreducible FSTD then the Shannon cover is unique.

For an FSTD we can obtain the transition or adjacency matrix $T(G) = [t_{ij}]$ of size $|V(G)| \times |V(G)|$. The entries t_{ij} are the number of edges in G from state i to state j. The transition matrix can be used to determine the capacity of an (M,d,k) constraint.

Specifically $C(M,d,k) = \log_2 \lambda(T)$, where $\lambda(T)$ is the largest real eigenvalue of $T(G)$. An alternative procedure to finding the capacity of an (M,d,k) code is given in [44], based on the enumeration of all sequences satisfying the constraint. From this enumeration, a difference equation is defined and a characteristic equation obtained. For the (M,d,k) constraint the capacity is the logarithm of the largest real root of the equation $\lambda^{k+2} - \lambda^{k+1} - (M-1)\lambda^{k-d+1} + (M-1) = 0$.

The performance of an (M,d,k) code for a recording system can be evaluated using the storage density D in (bits/unit area) and the error probability P_e (bit error rate). The code design problem is to increase the rate at which symbols are stored while satisfying the (d,k) constraints and keeping a low error probability. The rate R of the code is defined as the number of bits per symbol transmitted by the code. For a fixed rate code that encodes p binary digits into q channel symbols the rate is $R = p/q$. The capacity C the largest rate of any code satisfying the (d,k) constraints, i.e., $R \leq C$. The efficiency of the code is $E = R/C$. The density D can be calculated using $D = b/vTW$ bits/unit area, where b is number of bits stored in time T, v is rotating speed, W is the track width. Thus, letting $vW = 1$, we obtain the lineal bit density $D = b/T$ bits/sec of the system [33]. If every symbol in the channel has a duration T_s and if mod-M NRZI precoding is used, the length of the smallest recorded mark is given by $T_{min} = (d+1)T_s$ and the maximum recorded mark is $T_{max} = (k+1)T_s$. Using these definitions the lineal bit density in bits/T_{min} can be expressed by $D = \frac{(d+1)R}{T_{min}}$ bits/T_{min}. Note that an increase in d increases D, however it also increases the length of T_{min}, which means less marks can be written on the disk. If we use $T_{min} = (d+1)T_s$ we obtain an alternative formula for the density in bits/sec, $D = \frac{R}{T_s}$ bits/sec. Therefore, the only two ways to increase D are to increase the code rate R or decrease the symbol length T_s.

The encoder maps a p-block of information bits into a q-block sequence satisfying the RLL code constraints. The mapping may be a function of other parameters besides the p bits, such as the current state of the channel or even a set of future input tags as in the case of look-ahead codes [23, 39]. An encoder is called a block-encoder if only the p bits being

encoded are required for encoding. This type of encoder has only one state in its FSTD. The decoder does the reverse operation. It maps a q-block of channel symbols into a p-block of information bits. Just like the encoder, the decoder can also be a state-dependent decoder. In a state-dependent decoder the decision on the decoded bits is based not only on a set of received bits, but also on the state of the channel. The problem with this type of decoder is that if an error occurs, then the decoder can lose track of the correct state information and infinite error propagation can occur. Therefore, a decoder with a bounded time interval for errors given a single symbol error is desired. Such a decoder can be implemented with a sliding block decoder (SBD), which uses a window of $m+a+1$ q-blocks to generate a p-block of information bits. The window parameters m, a are the memory and anticipation defined above. A special type of SBD decoder is a block-decoder, one where $m = a = 0$. As with the block-encoder, a block-decoder will make a decision on the p input bits based solely on a q-block of received symbols.

2.4 Detection

Symbol detection is an important component of a communication system. The detector used can dramatically influence the probability of error in the system. Figure 10 shows a block diagram of a recording system. We can see that detection is performed immediately after the readout of the channel signal. Traditionally, there have been two types of detectors: symbol-by-symbol and sequence detectors. The first type estimates on one symbol at a time. The second type estimates a group of symbols all at once. Here, we introduce and briefly describe some of the detectors that are used optical recording systems.

2.4.1 Peak Detector

Peak detection was the most widely used detector of RLL constrained sequences in the early stages of magnetic and optical recording. Figure 11 shows the basic diagram of a gated peak detector. The top portion of the diagram represents the first line of processing in

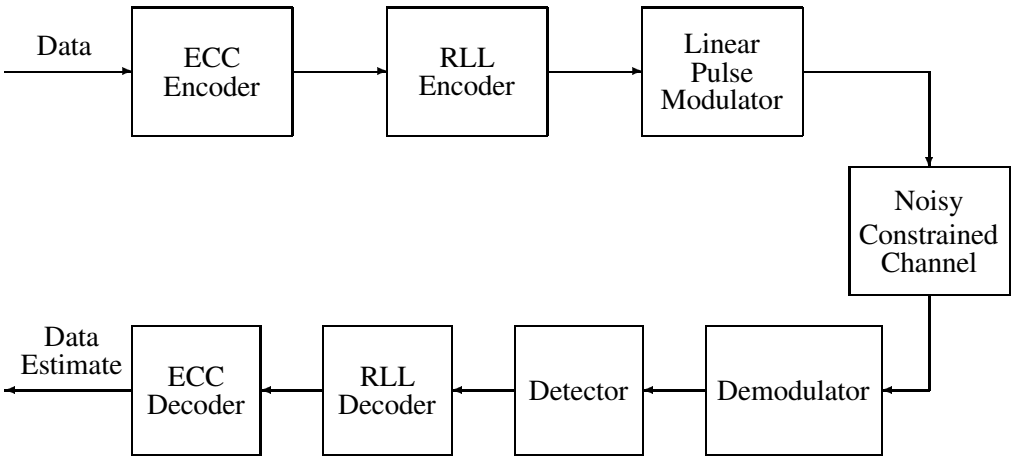


Figure 10: A complete model of an ECC-RLL concatenated recording system.

which the zero crossings of the first derivative of the signal are found. The bottom portion finds the peaks in the signal that exceed a specified threshold. A peak, which marks a transition in the signal, can only occur if there is a zero crossing in the first derivative. However, we can have zero crossings due to noise. These usually occur when the signal is of small amplitude and the amplitude threshold is not exceeded. Hence, a zero crossing without the threshold being exceeded is discarded as a crossing caused by noise. If both a zero crossing and a threshold crossing occur then a peak is found, which means there was a transition in the binary RLL signal. At the output of the detector peaks are used to mark that a transition in the signal occurs. These peaks are of alternating signs. The main drawback of peak detection is that it can only be used for low densities and for small noise variances. If these conditions change, i.e., if there is an increase in ISI or a decrease in SNR, then the performance of a peak detector is degraded. There are modifications to peak detection, such as pulse slimmers [4], that have been used in order to reduce the effects of ISI on peak detection. However these techniques have their own drawbacks, such as noise enhancement.

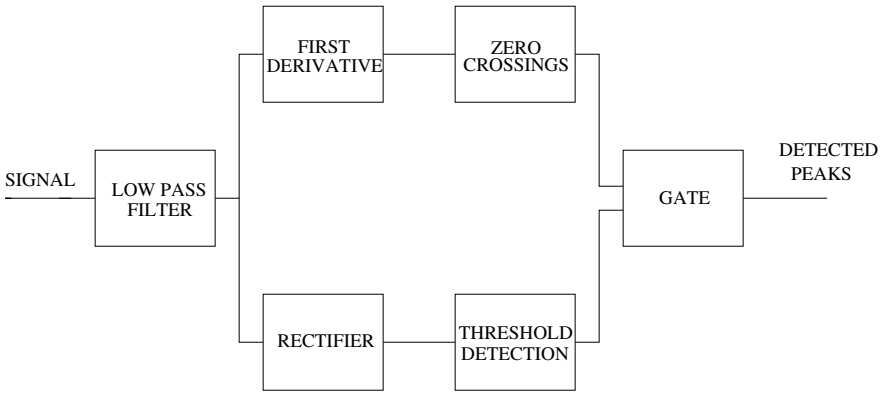


Figure 11: Diagram for peak detection

2.4.2 Zero-Forcing Linear Equalizer

A zero forcing linear equalizer (ZF-LE) is a filter whose purpose is to eliminate the ISI introduced by the channel. By removing the ISI, a simple slicer can be used for symbol-by-symbol detection of the signal. An alternative is to use the equalizer as a front end to a receiver and then apply a sequence detection method to the output of the equalizer. In this approach, the equalizer removes the ISI, and the sequence detector deals is used to decode the error-control code used in the system. This approach is the one taken for detection of the signal in ML CD-R/RW systems. The design of a ZF-LE is simple if we know the channel model. Assuming the channel frequency response is $H(f)$, then the frequency response of the ZF-LE will be $H(f)^{-1}$. The goal is that the frequency response of the combined channel-equalizer system be equal to one over the range of frequencies of interest. The main disadvantage of this type of equalizer is that it enhances additive white Gaussian noise. However, in ML CD-R/RW systems this problem does not manifest itself, since most of the noise in this case is media noise which is shaped by the channel frequency spectrum.

2.4.3 Viterbi Detection

This algorithm is used for maximum-likelihood sequence detection. It uses dynamic programming and was first introduced in [49]. The algorithm is used when the signal can be

modeled by a finite-state machine and the noise component in each sample is independent. The state transitions in the FSM are represented by a trellis diagram. The trellis represents the possible progression of states over time. It consists of states and branches. A branch is a transition from one state at time k to another state at time $k+1$. A path through the trellis is a collection of branches followed in order to get from state i at time k to state j at time $k+d$. Then, a path through the trellis corresponds to a possible transmitted sequence. A metric is applied to each branch that calculates the cost involved in taking that path at that particular instance in time. Each path in the trellis has its associated cost. The algorithm then selects as the most likely transmitted sequence the one with the smallest cost (metric). This type of detection is used in newer optical recording systems. Its main drawback is that the output sequence is a hard decision. There is no soft-information that tells us how certain we are about the decision being made.

2.4.4 Iterative Detection

Iterative detection is a recent development. It uses soft-input soft-output (SISO) detectors to obtain estimates of received sequences from concatenated codes. Its main advantage is that the detection process can be done more than once using information gained from previous iterations of the detection process. This can be done because of the soft-input soft-output characteristic of the detector. The idea is to use the information gained from the previous decoding iteration to decode in the current iteration. Then, the information gained from decoding the current iteration is used to decode in the next iteration. This process is repeated (iterated) for a finite number of times. The use of the information gained in the preceding steps allows us to obtain a better estimate of the signal.

The most widely used SISO detector in iterative detection is the Bahl, Cocke, Jelinek and Raviv (BCJR) algorithm, first introduced in [3]. This algorithm computes the *a posteriori* probabilities (APP) of the channel inputs given the channel output, channel estimates

and *a priori* probabilities on the channel inputs. It assumes the channel inputs are independent. The fact that it uses the *a priori* information at its input to improve the quality of its output and that it computes soft symbol estimates, in the form of the APP, make it suitable for iterative detection. Its main drawback is that the per-symbol computational complexity increases exponentially with the channel memory. This makes it inefficient for channels with a long impulse response.

Iterative detection is usually employed for Turbo coded detection. In fact, it is the sequence detector used in the ML CD-R/RW system that employs Turbo codes. It is a good alternative to hard decision algorithms, like Viterbi detection, when concatenated codes are used.

2.5 Conclusion

We have discussed the methods used to increase capacity in binary optical recording systems. In particular, we discussed how the laser wavelength and numerical aperture of the system are used to increase capacity. We introduced technologies that approach the increase in capacity differently. The goal of these technologies is to increase capacity by encoding signals with more than two amplitudes. Therefore, more bits can be encoded in the signal amplitude. One of these technologies uses constrained sequences. Therefore, an introduction to the design of M-ary constrained sequences was given. Finally, we discussed the detection issue and gave an introduction to symbol-by-symbol and sequence detection algorithms commonly used in optical recording.

CHAPTER 3

CHANNEL MODEL AND CAPACITY CALCULATIONS

Before attempting to transmit a signal over a communication channel, it is of interest to understand the fundamental limits of this channel, in particular how much information we can reliably transmit over the channel in question. This measure is commonly referred to as the "capacity" of the channel. Once this measure is known, we can attempt to transmit information at a rate that approaches this capacity.

Here, we calculate the fundamental limit for signaling on the optical recording channel. To calculate channel capacity we need to make specific assumptions about the channel. To that end, the mathematical model for our channel will be presented. In particular, we define the channel frequency response, the noise model and the frequency response of filtering operations at the receiver. Then, we will compute the "waterfilling" capacity of this channel model. The capacity is calculated for a model in which a continuous-time signal can be recorded in the channel, as this provides the maximum value achievable. Then a more realistic model, where the recorded signal has to vary over discrete intervals, is presented. We calculate capacity for this model and compare it to that of the continuous-time model. The models are kept general by making as few assumptions as possible.

3.1 "Waterfilling" for Capacity Calculation

The "waterfilling" technique is used to calculate the capacity of a given model. Let $N(f)$ be the noise PSD, $H(f)$ be the channel frequency response and x_k the channel input. It is shown in [19] that to compute the waterfilling capacity the channel may be seen as a set of narrowband, independent channels. We assume that the input is power constrained. The

power constraint for the input waveform is given by

$$\sum_i \bar{x_i^2} \leq ST. \tag{2}$$

Then, the capacity of the parallel combination normalized to capacity per unit time is

$$C_T = \sum_{i \in I_B} \frac{1}{2T} \log \frac{|H(i/T)|^2 B}{N(i/T)}, \tag{3}$$

where I_B is the set of i for which $N(i/T)/|H(i/T)|^2 \leq B$ and B is the solution to the equation

$$S = \frac{1}{T} \sum_{i \in I_B} \left[B - \frac{N(i/T)}{|H(i/T)|^2} \right]. \tag{4}$$

The amount of energy needed in each channel to achieve capacity is given by

$$\bar{x_i^2} = \begin{cases} B - \frac{N(i/T)}{|H(i/T)|^2} & i \in I_B \\ 0 & i \notin I_B \end{cases}. \tag{5}$$

If we let $T \to \infty$, in the limit, the summations in (3) and (4) become Riemman integrals and we then have

$$C = \lim_{T \to \infty} C_T = \int_{f \in F_B} \frac{1}{2} \log \left[\frac{|H(f)|^2 B}{N(f)} \right] df, \tag{6}$$

where F_B is the range of f for which $N(f)/|H(f)|^2 \leq B$, and B is the solution to

$$S = \int_{f \in F_B} \left[B - \frac{N(f)}{|H(f)|^2} \right] df. \tag{7}$$

The power spectral density that achieves capacity for the input ensemble is given by

$$S_x(f) = \begin{cases} B - \frac{N(f)}{|H(f)|^2} & f \in F_B \\ 0 & f \notin F_B \end{cases}. \tag{8}$$

The interpretation of the above equations is observed in Figure 12. The power S is given by the total area of the shaded regions. The power spectral density (PSD), at a particular frequency (f), is given by the height of the shaded region for that f. The power can be distributed in a way that achieves capacity, by allocating power to the frequencies in the channel where the ratio of the noise PSD to the square magnitude of the channel frequency response is smallest.

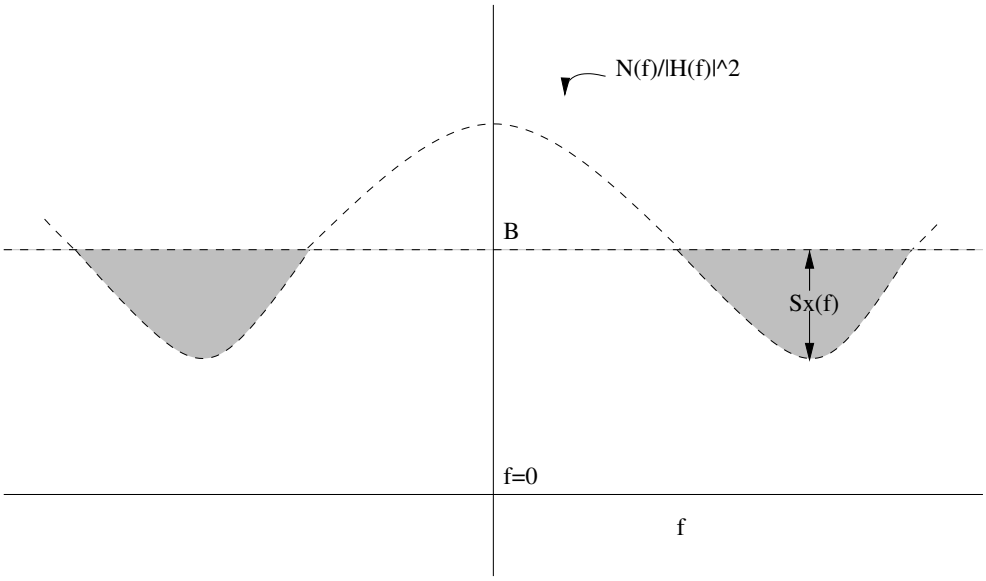


Figure 12: Waterfilling input power distribution.

3.2 Continuous Time Model

Figure 13 shows the model for the continuous-time optical recording channel. It consists of the channel response $h_c(t)$, the noise signal $n(t)$, a receive filter $h_r(t)$, and an input signal $x(t)$. The signal $y(t)$ is the signal before the receive filter and is given by $y(t) = x(t) * h_c(t) + n(t)$, where $*$ denotes continuous-time convolution. The received signal $r(t)$ is given by $r(t) = y(t) * h_r(t)$. The input signal $x(t)$ is assumed to be a continuous-time, continuous-valued signal with a power constraint. Therefore, it satisfies

$$E\left[\frac{1}{T}\int_{-T/2}^{T/2} x^2(t)dt\right] \leq S, \tag{9}$$

where $E[\cdot]$ is the expectation operator, S is the power constraint, and T is the symbol period.

We have already specified the frequency response $H_c(f)$ for the channel $h_c(t)$. We will use the square of (1). The receive filter $h_r(t)$, or its frequency response $H_r(f)$, does not have to be specified to calculate capacity. All we need is for the filter to be invertible. The noise signal $n(t)$ consists of a combination of electronic noise and media noise.

The electronic noise is a white Gaussian random process that adds to the output of the recording channel. The electronic noise models the noise in the system due to the electronic

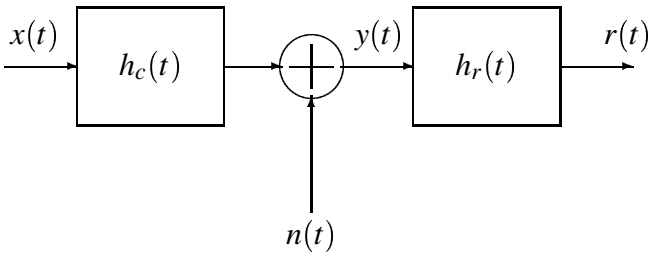


Figure 13: Continuous-time model for the optical recording channel.

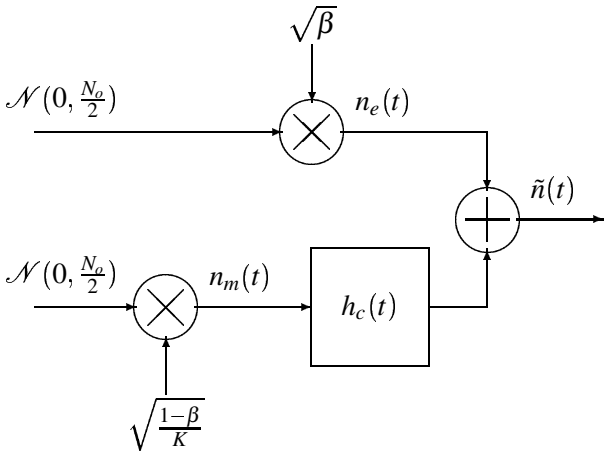


Figure 14: Model for noise signal $\tilde{n}(t)$.

components in the detector. Amplifying circuits, motors and other components contribute to noise that creeps into the readout signal. Additive white Gaussian noise is a good model for this type of noise.

The media noise is caused by imperfections in the written media. In particular, transition jitter is the main source of noise from the recorded media. Transition jitter refers to small variations in the lengths of recorded marks with respect to their ideal length. This type of noise can be modeled by a Gaussian random process whose spectrum is shaped by $|H_c(f)|^2$, the channel frequency response [5]. At high SNR, when the contribution from electronic noise is negligible, this source of noise is the dominant source. The factor β gives the proportion of the electronic noise in the noise signal. Then the total noise $n(t)$ is

zero-mean, band-limited noise with power spectral density

$$S_n(f) = \frac{1-\beta}{K}\frac{N_o}{2}|H_c(f)|^2 + \beta\frac{N_o}{2}, \tag{10}$$

where $K = (\int_{-\infty}^{\infty}|H_c(f)|^2 df)/2W$ and W is the bandwidth of the channel. Figure 14 gives a detailed model of the noise signal.

3.3 Model Assumptions

The interpretation of "waterfilling" given in Figure 12 is a general interpretation. For the specific case we are analyzing, $H_c(f)$ as given in (1) and $S_n(f)$ is given in (10). We see that we have only one "bowl" to fill. The channel is also constrained to be zero for frequencies $f \geq |f_c|$. Therefore, the correct interpretation of the water level B is more along the lines of that shown in Figure 15. Expressing B mathematically, we have

$$B = \begin{cases} \frac{N(f)}{|H(f)|^2} & \text{if } B \leq B_{f_c} \\ B_{f_c} + \frac{N_o}{2}M & \text{if } B > B_{f_c} \end{cases}, \tag{11}$$

where $B_{f_c} = N(f_c)/|H(f_c)|^2$. With this formula for B and using (6) and (7), we can calculate the curve of signal-to-noise ratio (SNR) vs. capacity for the channel model in Figure 13. This is accomplished by numerical integration of (6) and (7). An attempt was made to find a closed form solution. However, even for the simple Gaussian channel no closed form solution could be found.

The capacity calculated with (6) will be in bits/second or in bits if a normalized frequency $f_n = f/f_c$ is used, since f_n is a dimensionless quantity. If we use this normalization, then $f_n \in [-1,1]$. This is because we are interested only in the range $f \in [-f_c, f_c]$, where the channel frequency response is nonzero. However, it is of interest to be able to compare results across different types of systems. Therefore, we would like to have our capacity calculations in units that can be used to compare different recording systems. Such a unit would be bits/*MF* since it tells us how many bits can be stored in the smallest readable mark in a particular system. To express capacity in bits/*MF*, we use the fact

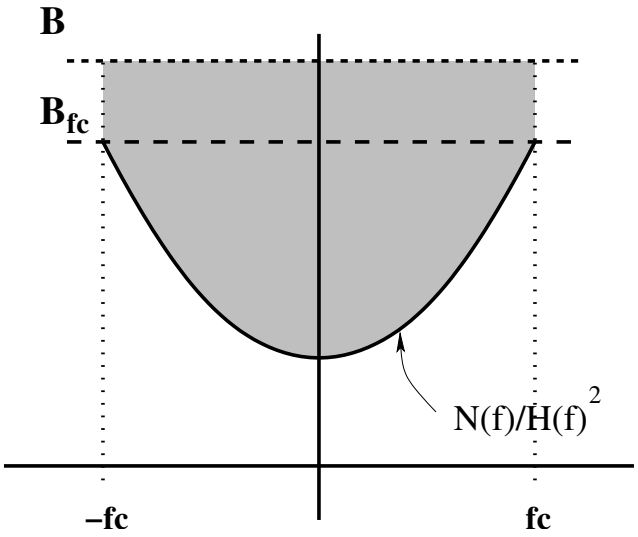


Figure 15: Waterfilling input power distribution for optical recording channel.

that $f_c = 2NA/\lambda$ m^{-1} and $l_{MF} = \lambda/4NA$ m/MF is the approximate length in meters of one minimum feature. Then, multiplying capacity in bits C, calculated using (6), by l_{MF} and f_c we get $C_{MF} = \frac{C}{2}\frac{\text{bits}}{MF}$. We will use these same units for capacity in the following chapters. Typical CD-RW parameters are used in our analysis. Therefore, the laser wavelength is $\lambda = 780$ nm and the numerical aperture, $NA = 0.5$.

For comparison purposes it is necessary to specify what we mean by SNR, in particular, at which point in the system will SNR be measured. Once again, we are interested in comparing results across a variety of systems. Therefore, it is of interest to use the SNR at a point where the effect of the system's channel frequency response has already been taken into account. We will therefore use the SNR at the detector or in Figure 13, the SNR of $r(t)$. Calculating the SNR at this point can be a complicated process. To avoid this, we use an upper bound on the SNR obtained by the application of the Schwarz Inequality to the SNR formula. This allows us to separate the SNR term into the SNR at the input of the channel multiplied by the square magnitude response of the channel. This procedure lower bounds the capacity of the channel, which is what we are after.

When we introduced our model in Figure 13, we commented that knowing $H_r(f)$ is not

really important. This is true since it will not affect the capacity of the system, as long as $H_r(f)$ is invertible. Consider a modified channel $\tilde{H}(f) = H_c(f)H_r(f)$ and modified noise signal $\tilde{N}(f) = N(f)|H_r(f)|^2$, the result of filtering $H_c(f)$ and $N(f)$ with the receive filter $H_r(f)$. We notice that $\frac{|\tilde{H}(f)|^2}{\tilde{N}(f)} = \frac{|H(f)|^2}{N(f)}$ as long as $H_r(f)$ is invertible since in that case we can cancel those terms in the numerator and denominator.

The irrelevance of the $H_r(f)$ for the channel capacity can also be explained with the data processing theorem from information theory [13]. From this theorem, any data processing applied to a signal cannot increase the information in the signal; it can only decrease or leave the same information in the signal. The latter occurs only if the processing applied is invertible, which is why we require $H_r(f)$ to be invertible. If this is the case, we are not changing the information contained in the signal and the capacity stays the same.

The channel model used in the following sections is the square of the MTF found in (1). However, we also analyzed the performance of another channel model, based on a Gaussian approximation of the MTF^2. The channel frequency response for this channel is given by

$$H(f) = \begin{cases} e^{-\frac{f^2}{2\sigma^2}} & |f| < 1 \\ 0 & \text{otherwise} \end{cases}, \tag{12}$$

where f is the normalized frequency and σ is calculated so that the total power under the Gaussian curve is the same as the total power under the MTF^2 channel response. Figure 16 compares the two frequency responses. The MTF^2 model was used because it more accurately represents the channel frequency response. The tails are longer than those obtained with the Gaussian channel. Also, less of the energy is concentrated at lower frequencies.

3.4 Capacity Results for the Continuous Time Model

In this section we show curves of SNR vs. capacity for the continuous-time model. We consider two parameters that affect channel behavior:, β, the percentage of white Gaussian noise in the noise signal $n(t)$ and the channel usage α, which determines how much of the

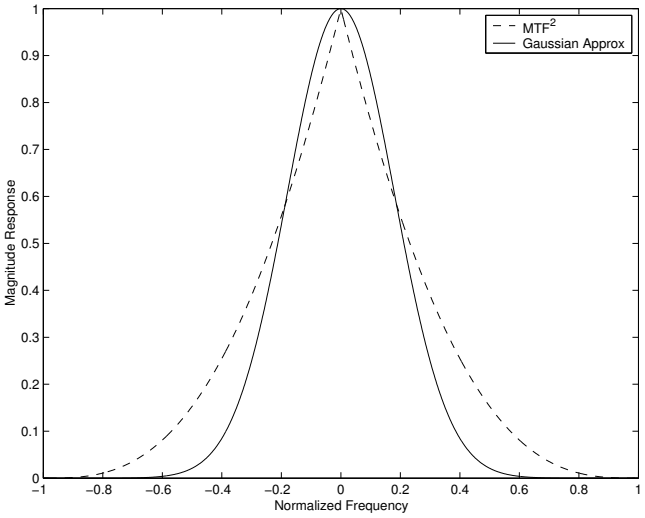


Figure 16: MTF^2 and Gaussian approximation.

channel bandwidth will be used by the signal. Therefore, its value must lie in the range $[0,1]$. A value of $\alpha = 1$ means we are using the whole channel bandwidth, while $\alpha = 0.5$ means we use only half of the channel bandwidth to allocate signal power. In other words, if the channel usage is α, we use only the frequencies $f_n \in [-\alpha, \alpha]$ for signaling. This can be accomplished by shaping the PSD of the input signal to the channel.

Figure 17 shows the curves of capacity vs. SNR for different values of β for both the Gaussian approximation channel and the MTF^2 channel. For low SNR values (SNR$<$15 dB) capacity is higher for larger values of β. However, in the high SNR domain (SNR$>$15 dB) capacity is higher for lower values of β. From the waterfilling point of view, a lower value of β means the "bowl" is more shallow. In the extreme case of $\beta = 0$, we have no "bowl". Instead, $\frac{N(f)}{|H(f)|^2} = 1/K$, with K as defined in Section 3.2. This extreme case implies that the noise signal consists solely of media noise. From Figure 17 we see that in the low SNR region electronic noise is the limiting factor for capacity. In the high SNR region, where most systems operate, media noise is the dominant source of errors.

Figure 18 shows the capacity curves for different values of α and two different values of β. From these curves we can see that the best approach to record signals is to use the

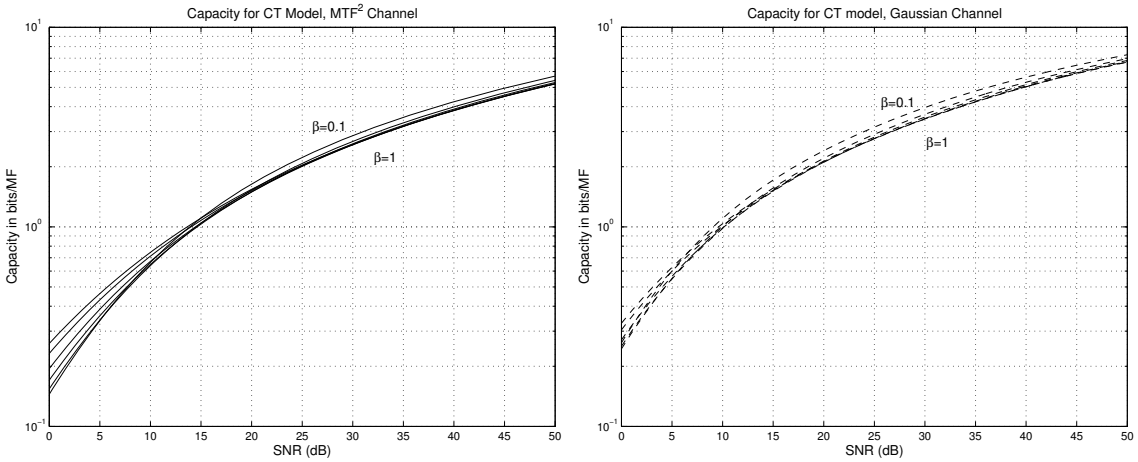


Figure 17: SNR vs. capacity for varying β.

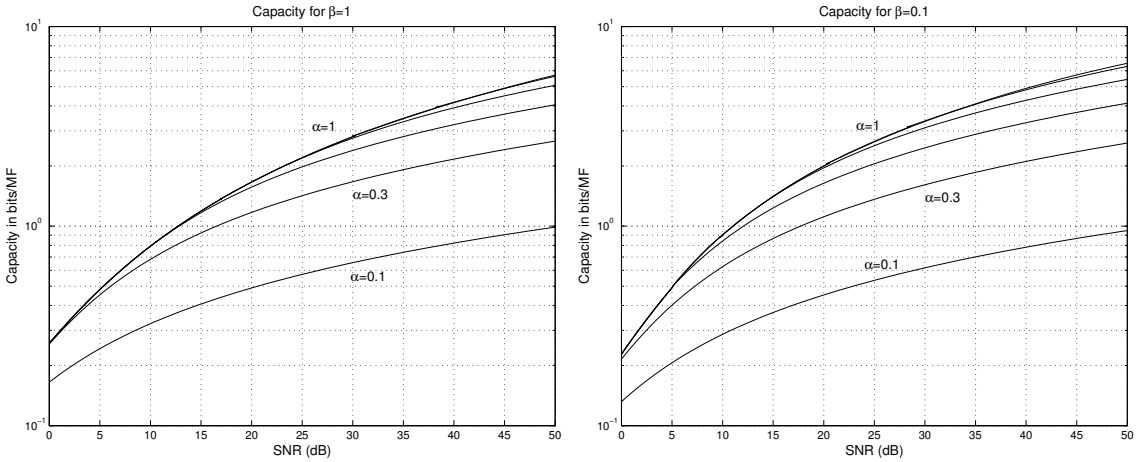


Figure 18: SNR vs. capacity for different α.

whole channel bandwidth available, since the curve for $\alpha = 1$ is the curve with the highest capacity. The lower curve represents $\alpha = 0.1$. Then, there is an increase of 0.2 in the value of α for each of the following curves except for the highest curve, which is the one representing $\alpha = 1$. Notice that, when $\beta = 1$, in the low SNR range the capacity is the same for all curves except for $\alpha = 0.1$. When $\beta = 0.1$, this is true for all curves with $\alpha \geq 0.5$. This can be explained by the fact that at those low SNR levels we can not use the whole channel bandwidth because the noise level at larger values of α is too high. The change in behavior from $\beta = 1$ to $\beta = 0.1$ can be attributed to the fact that the "bowl" for $\beta = 0.1$ is

shallower. Therefore, we can use more of the channel bandwidth at low SNR and capacity is larger for the curves that use more channel bandwidth.

3.5 The Discrete Time Channel

Until now, we have dealt with a continuous-time channel model in which a continuous-time signal is transmitted. However, the properties of the optical recording channel do not allow for such a signal to be written on the disc. Instead, a discrete signal is used. Therefore, we must modify our model to one that has a closer resemblance to the optical recording channel. Figure 19 shows such a model. The new channel is a discrete-time channel in which marks of length T are written on the disc. In particular, the new model has a discrete-time, discrete-valued multilevel input and a discrete-time continuous-valued output. Its input is the sequence $\{a_k\}$, $(a_k \in \{\pm 1, \pm 3, \ldots, \pm(M-1)\})$ and its output is given by

$$y(t) = \left(\sum_k a_k p_T(t - kT) \right) * h_c(t) + \tilde{n}(t) \tag{13}$$

$$r(t) = y(t) * h_r(t) \tag{14}$$

$$r_k = r(kT), \tag{15}$$

where T is the duration of one channel symbol and $p_T(t)$ is the write pulse. In our model, we have selected a square write pulse for simplicity.

The formulas used to calculate the capacity of the continuous-time model do not apply for this type of channel. New formulas must be derived, that take into account the new constraints. Both [46] and [10] deal with the capacity of discrete-time channels. For our analysis, we will use formulas (53a), (53d), and (60) in [10], since these can be used directly in our problem. The formulas are

$$C = \frac{1}{4W} \int_{-W}^{W} df \max(0, \log_2 \frac{B|H_D(f)|^2}{N_D(f)}) \tag{16}$$

$$S_D = \frac{1}{2W} \int_{-W}^{W} df \max(0, B - \frac{N_D(f)}{|H_D(f)|^2}) \tag{17}$$

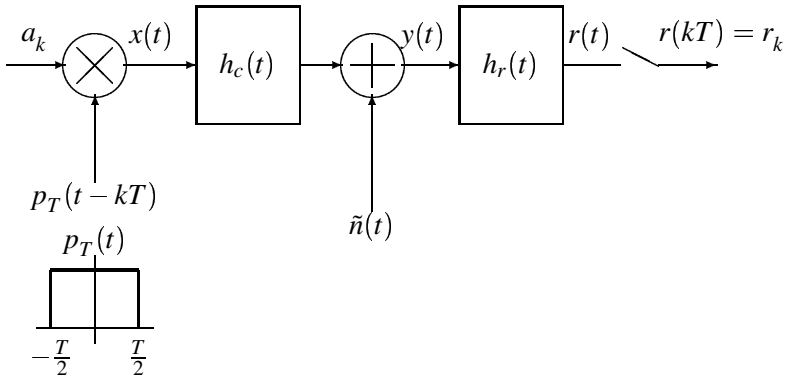


Figure 19: The discrete-time channel model.

$$N_D(f) = \frac{1}{2W}N(f) \tag{18}$$

$$H_D(f) = H(f), \tag{19}$$

where W is the inverse of the pulse shape duration. The parameter $W = 1/T$, is also the sampling rate of our discrete-time channel model. Note that the W used here is different from the W used in Section 3.2. The one in Section 3.2 refers to the channel bandwidth, which we have assumed to be f_c. Whenever needed we will specify what definition of W we are referring to.

There are some changes to these formulas that need to be taken into account. Notice from Figure 19 that the overall continuous-time channel is now a combination of the pulse shape, the optical recording channel, and the receive filter. Therefore, our $H(f)$ will change. We define a new transfer function $\tilde{H}(f)$. The mathematical expression for this function is $\tilde{H}(f) = P_T(f)H_c(f)H_r(f)$, where $P_T(f)$ is the Fourier transform of the pulse shape $p_T(t)$ (see Figure 19). This Fourier transform is given by

$$P_T(f) = \frac{T}{2}\text{sinc}(\pi fT). \tag{20}$$

Therefore, $\tilde{H}(f) = \frac{T}{2}\text{sinc}(\pi fT)H_c(f)H_r(f)$ is the CT transfer function for the new channel and $H_D(f) = \tilde{H}(f)$. Also, $N_D(f) = S_n(f)|H_r(f)|^2$ is the noise PSD that needs to be used. Simple substitution in equations (16) and (17) shows that if $|H_r(f)|^2$ is invertible it will

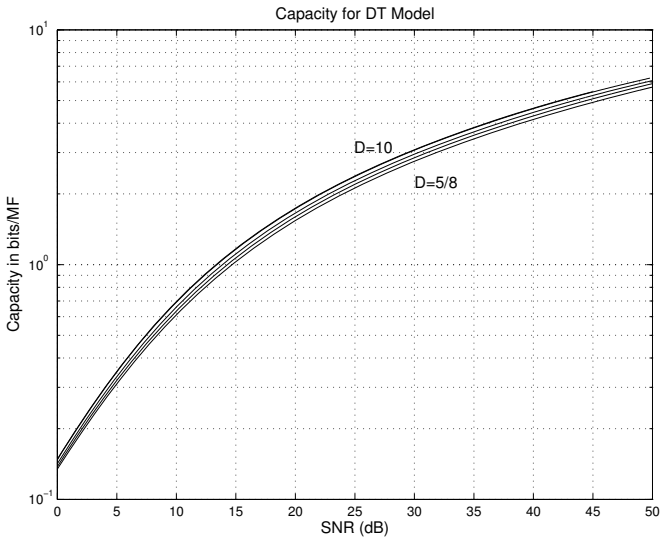


Figure 20: SNR vs. capacity for discrete-time channel.

cancel out from those equations. Therefore, an invertible receive filter leaves channel capacity unaffected. Notice that we keep the power constraint on the input signal as given in (9), where $x(t)$ is now the term in parentheses in (13). We can evaluate these formulas, using numerical integration, to obtain expressions for capacity and SNR of the discrete-time channel.

As can be seen from the formulas, the pulse length T affects the capacity curves. It is of interest to see how changing T changes the curves. In all our previous capacity plots, the capacity units have been bits/*MF*, where *MF* stands for minimum feature, which is equal to $\lambda/4NA$. We have also defined l_{MF} as the signal whose length is equal to one minimum feature. Then, the value $D = \frac{l_{MF}}{l_m}$ is a variable that gives us the number of channel uses per minimum feature. Here l_m, the length of a mark in meters, given by $l_m = vT$, where v is the spinning velocity of the disc and T is as defined above. As T decreases, there are more channel uses per minimum feature. Figure 20 shows SNR vs. capacity curves for values of $D = \{5/8, 3/4, 1, 3, 10\}$. It can be seen that as T decreases (D increases), the capacity increases. The line for $D = 3$ and the one for $D = 10$ are indistinguishable, since there is very little variation in capacity between those two values. In fact, the capacity for both

these values is essentially that of the continuous-time system.

When comparing these results to the continuous-time case we see that there is very little to gain by decreasing the mark size. In Figure 20, we assume that the length of the mark can actually be made smaller than a minimum feature $D = \{3, 10\}$. This is not the case in real life. However, even if we could make a mark smaller than the minimum feature, we notice that there is little to be gained by doing so. In particular, we notice that a value of $D = 5/8$ gives a capacity curve close enough to that the $D = 1$ system, while maintaining the mark size large enough so that ISI in the channel can be easily handled.

3.6 Conclusion

In this chapter we have given a simplified model for the optical recording channel (ORC). We first introduced a continuous-time model, which was later extended to the discrete-time case. Capacity for both models was calculated using the "waterfilling" method. The influence of system parameters, such as β, α, and D, on the capacity of the system was analyzed in plots of SNR vs. capacity. Through these plots, a relationship between capacity and each of the system parameters was found.

CHAPTER 4

UNCODED AND RLL CODED PERFORMANCE

In Chapter 3, we introduced a model for the optical recording channel (ORC). In this chapter we extend that model and use it to study the performance of uncoded M-ary fixed-length signaling and RLL coded M-ary signaling over the ORC. We start by specifying the receive filter $h_r(t)$. We then use spectral factorization on the overall system response to give a simplified discrete-time model for the model in Figure 19 and for an RLL coded channel. The simplified models will give us discrete-time impulse responses that can be used to define trellises for Viterbi detection of recorded signals in the ORC. Then performance of both systems is evaluated and plots of SNR vs. error probability are presented.

4.1 Sufficient-Statistics Discrete-Time Channel Model

In this section we describe the process that converts a continuous-time optical recording channel model with media noise into an equivalent sufficient-statistics discrete-time (SSDT) channel model. The SSDT channel model is required to obtain the simplest detector structure for a given optimal detection criterion. It is well known that to obtain an SSDT channel model [40], the noise is first whitened and then a matched filter is placed before the symbol sampling occurs. Therefore, we use the cascade of a noise whitener and a matched filter as $h_r(t)$ in Figure 19. Because of the matched filter, the noise samples at the sampler output are not white, but in fact have a real power spectral density $S_{h,n}(e^{jfT})$, also known as the folded spectrum, given by

$$S_{h,n}(e^{jfT}) = \frac{1}{T}\sum_{m=-\infty}^{\infty} \frac{\left|H\left(j\left(f+\frac{m}{T}\right)\right)\right|^2}{S_n\left(j\left(f+\frac{m}{T}\right)\right)}. \tag{21}$$

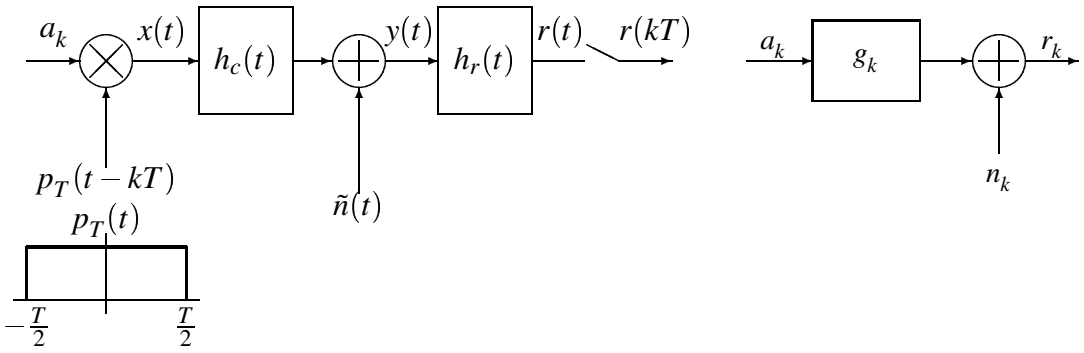


Figure 21: Discrete-time channel model for fixed-length marks.

Since $S_{h,n}(e^{jfT})$ is always real, we can use minimum-phase spectral factorization to write

$$S_{h,n}(z) = A_{h,n}^2 G(z) G^*(1/z^*). \tag{22}$$

Both $G(z)$ and $G^*(1/z^*)$ are noise whiteners. However, choosing the whitener to be anti-causal is more common in practice. Thus, assuming that the sampler output is filtered by $G^*(1/z^*)$, the equivalent SSDT model consists of the input sequence $\{a_k\}$, the discrete-time channel taps $\{g_k\}$, and an additive white Gaussian noise signal $\{n_k\}$ with $S_N(f) = 1/A_{h,n}^2$. Figure 21 shows the complete system model for the uncoded (fixed-length) mark channel and its SSDT channel model. This model applies for both uncoded and error-control coded channels, just as long as the marks are fixed-length. Therefore, it can also be used to represent the ML CD-RW channel.

For our model of the ORC, the SSDT channel taps are dependent upon two parameters: the number of channel uses per minimum feature D and the percentage of white Gaussian noise in the noise signal β, which were defined in Chapter 3. As D increases, there is more intersymbol interference (ISI) in the channel. Since the channel model $\{g_k\}$ is monic and causal, this means that the magnitude of the non-unitary taps in the channel response increase with an increase in D. This makes correct signaling harder. As β decreases, that is, the percentage of electronic noise in the channel decreases, we find that there is a reduction in the ISI in the channel. Therefore, the ISI in the channel is proportional to both D and β.

Table 2: The percentage of three neighboring channel taps with $\beta = 0.2$.

D	g_1	g_2	g_3
1/4	10.8%	0.4%	0.14%
1/3	16.02%	0.26%	0.18%
1/2	36.58%	2.48%	0.74%
5/8	63.59%	2.13%	0.32%

Table 3: The percentage of three neighboring channel taps with $D = 0.5$.

β	g_1	g_2	g_3
0.1	25.54%	2.41%	0.51%
0.2	36.58%	2.48%	0.74%
0.5	59.02%	5.28%	1.28%

The behavior of the channel taps for an uncoded channel can be seen in Tables 2 and 3.

Table 2 shows the percentage of channel taps $g_1,\ g_2$, and g_3 with respect to the first (unitary) channel tap (g_0), for different values of the lineal density D and a fixed value of β. We see that an increase in D causes the percentage of subsequent channel taps to increase. Therefore, more ISI is present in those channels. Table 3 is similar to Table 2. However, the lineal density D is now fixed and the value of β varies. Once again we see that an increase in β causes an increase in the ISI present in the channel, as shown by the percentages given for taps $g_1,\ g_2$, and g_3.

For uncoded signals, we use a Viterbi detector. The trellis for the Viterbi detector is based on the discrete-time impulse response $\{g_k\}$ found from the SSDT model. For the densities simulated in the results section, we have found that a four-tap $\{g_k\}$ gives a precise representation of the channel. The magnitude of the fourth tap is in general less than two percent that of the first tap. The total energy contained in those four taps is greater than ninety nine percent of the total channel energy. Therefore, the trellis will consist of M^3 states, and for the particular case of $M = 8$, the trellis will have 512 states.

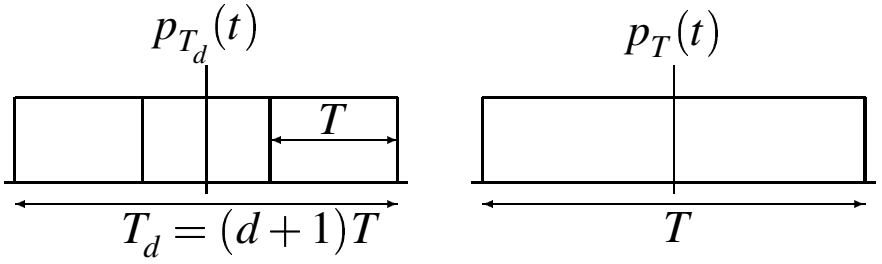


Figure 22: The pulse shape for RLL and fixed-length marks.

4.1.1 Performing Spectral Factorization

Once the folded spectrum $S_{h,n}(f)$ is obtained, there is a need to find a spectral factorization of it. To do this, a polynomial in powers of z whose frequency response is a good match to $S_{h,n}(f)$, in the mean squared error sense, is found. Once this polynomial is found, its roots are used to find g_k. This process assumes that $S_{h,n}(f)$ is an all-zeroes frequency response, that is, it has no poles. It was found that a reasonable approximation can be found as long as an adequate number of zeros is used for the approximation. The more zeros used, the longer the impulse response of g_k. Therefore, an adequate trade-off between accuracy and the length of g_k has to be found to avoid an increase in the complexity of the trellis. As described above, it was found that a channel impulse response with four taps provides a reasonable compromise.

The constant $A_{h,n}^2$ is equal to the geometric mean of the folded spectrum [28], or expressed mathematically

$$A_{h,n}^2 = \exp\{\int_{-\frac{1}{2}}^{\frac{1}{2}} \ln[S_{h,n}(\Omega)]d\Omega\}. \tag{23}$$

Since we know the magnitude response of the folded spectrum, we can easily evaluate this integral numerically.

4.2 Equivalent Channel for the RLL Coded Signal

Until now, we have analyzed the case of uncoded transmission over the optical recording channel. When the transmitted signal is RLL coded, the mark size now represents the size

of the smallest RLL coded mark. If we are using a $d = 2$ code, then the mark size will be that of a mark that contains three symbols. This means that the symbol length will be given by $l_s = l_m/(d+1)$. To account for this, our pulse shape $p_d(t)$ now consists of $d+1$ pulses joined together. Figure 22 shows the RLL pulse shape for $d = 2$ and compares it with the pulse shape for the uncoded case. In this figure we assume the uncoded mark is equal in size to the minimum mark of the RLL coded case. Notice that the value of T for the RLL case need not be equal to that of T for the uncoded case. Then, the pulse shape is specified by $p_d(t) = \sum_{i=0}^{d} p_T(t - iT)$, the frequency response of which is $P_d(f) = P_T(f) \sum_{i=0}^{d} e^{-j2\pi ifT}$. The right side of this last equation can be summed up to yield

$$P_d(f) = P_T(f) \frac{1 - e^{-j2\pi fT_d}}{1 - e^{-j2\pi fT}}, \tag{24}$$

where $T_d = (d+1)T$.

We now introduce the function $Q_{d+1}(f)$, a periodic function in f. Its period is $N/T_d = 1/T$ Hz and its mathematical expression is

$$Q_{d+1}(f) = \frac{\sin^2(\pi f T_d)}{\sin^2(\pi f T)}. \tag{25}$$

Then, using $Q_{d+1}(f)$, we have $|P_d(f)|^2 = Q_{d+1}(f)|P_T(f)|^2$ and after spectral folding we find that

$$S_{h_d,n}(f) = \frac{1}{d+1} \sum_{n=0}^{d} Q_{d+1}\left(\left(f + \frac{n}{T_d}\right)T\right) S_{h,n}\left(\left(f + \frac{n}{T_d}\right)T\right). \tag{26}$$

The folded spectrum $S_{h_d,n}(f)$ depends on the folded spectrum of the fixed-length mark channel $S_{h,n}(f)$ with mark length T and on the function $Q_{d+1}(f)$. This $S_{h_d,n}(f)$ can then be used to find the discrete-time equivalent channel model by performing spectral factorization [5].

The final discrete-time equivalent channel model is similar to the one in Figure 21. The main difference between both models is that the sampling clock in the RLL coded model runs $d+1$ times faster than the clock in the uncoded M-ary case. This is a direct result of the fact that the pulse shape for the RLL case is a summation of fixed-length pulses. Figure 23 shows the complete models for the RLL coded system.

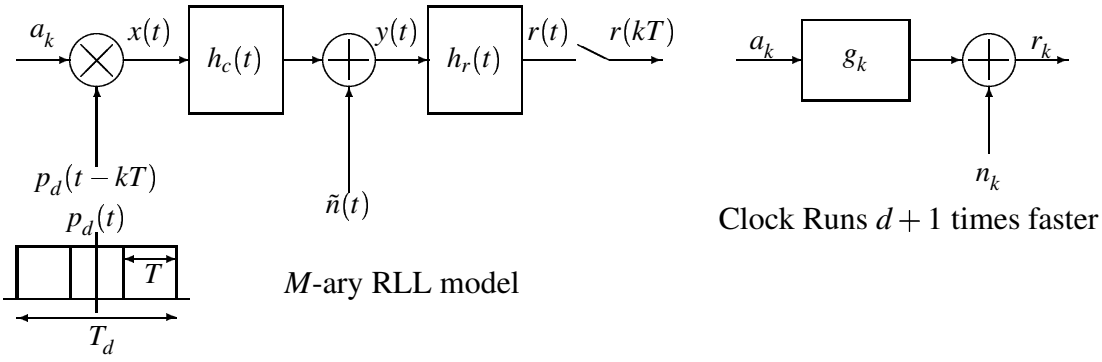


Figure 23: Discrete-time channel model for variable-length marks.

Table 4: Sample discrete-time equivalent channels for RLL coded case.

D	β	g_k
0.2	0.05	1 0 0 0.0529 0 0 -0.0001
0.4	0.05	1 0 0 0.1803 0 0 -0.0101
0.6	0.05	1 -0.0001 0.0001 0.3065 0 0 -0.05007

4.2.1 Performing Spectral Factorization

As in the fixed-length (uncoded) mark case, spectral factorization for the variable-length (RLL coded) mark case assumes that the frequency response is an all-zeroes frequency response. Then, a polynomial in powers of z whose frequency response is a good match, in the mean squared error sense, to the folded spectrum $S_{h_d,n}(f)$ is found. The roots of this polynomial are used to find g_k. The main difference between the uncoded and RLL coded case is in the length of the polynomial needed to obtain an adequate approximation. In general in the RLL coded case g_k will consist of seven taps, as opposed to four in the uncoded case. Some examples of channels in the RLL coded case are found in Table 4. The constant $A^2_{h,n}$ is calculated using (23).

4.3 M-ary RLL Codes Designed by Permutation

In our results, we will plot SNR vs. P_e, where P_e is the coded symbol error probability. Therefore, we need to compare only the read and written codeword (as opposed to comparing the read and written user bits). This allows us to perform simulations using highly

efficient (ninety nine percent efficient) permutation codes, since we don't actually need to perform encoding and decoding of user bits into the RLL sequence. We use codewords of runlength-limited codes with parameters $d = 2$, $k = 10$ and $M \in \{2,4,6,8\}$. The process by which we find the codewords is described in [14] and a short description is given here. Permutation codes were selected because they are fixed-length block codes whose codewords can be directly concatenated without requiring the use of merging bits.

We can calculate the capacity of the constraints of the (M,d,k) code with the equation $C = \log_2 \lambda$ bits/symbol, where λ is the largest real root of the characteristic equation $z^{k+2} - z^{k+1} - (M-1)z^{k-d+1} + (M-1) = 0$, see Section 2.3.1 for more details. The codeword will consist of phrases of length i, $i \in \{d+1,\ldots,k+1\}$. A phrase of length i is a concatenation of one non-zero symbol at the start of the phrase and $i-1$ zeros. We choose phrases with the non-zero symbol at the start of the phrase because of the following two reasons:

- We assume the channel rest state is the zero symbol.
- We use mod M precoding. Therefore, it is ideal for the non-zero symbol in the phrase to come at the beginning of the phrase. If this is true, all symbols in one particular phrase will have the same value.

With precoding the runlength of the phrases will become $(d+1,k+1)$. Then 1000 200 100000 300 are four phrases satisfying $d = 2$ and $k = 5$ constraints. Assuming $M = 4$, then after mod 4 precoding the four phrases become 1111 333 000000 333, where we assumed that the channel at rest is transmitting the '0' symbol.

We know specify how to select the number of phrases of length $i \in [d+1,k+1]$ in a codeword of length n, such that, the designed code asymptotically approaches capacity. Let X be a random variable describing the number of symbols in a phrase. Then, $P(X = i) = (M-1)2^{-iC}$, is the probability that X is equal to i [14]. Since the $P(X = i)$ are probabilities, $\sum_{i=d+1}^{k+1}(M-1)2^{-iC} = 1$. Let n_i be the number of phrases of length i in a codeword. Initially, assume that $n_i = \lceil p_i/p_s \rceil$,where $p_i = P(X = i)$, p_s is the smallest p_i, and $\lceil x \rceil$ denotes the

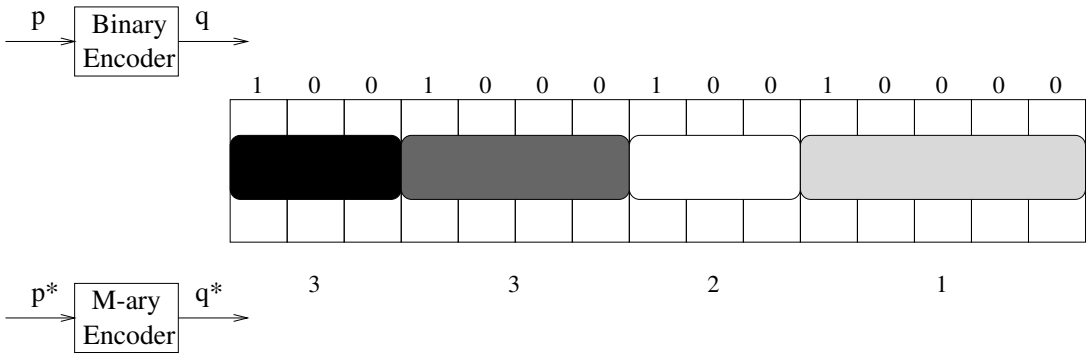


Figure 24: M-ary permutation code encoder.

smallest integer larger than x. This set of n_i would give a codeword of length $n = \sum_{i=d+1}^{k+1} n_i$. If this n is not the desired codeword length then we multiply the current values of n_i by the ratio of the desired codeword length to the current n and use the ceiling function to obtain an integer value for the new n_i. Let n_d denote the desired codeword length and $\hat{n}_i$ the number of phrases of length i that achieve that codeword length, we find the values for $\hat{n}_i$ with the formula $\hat{n}_i = \lceil \frac{n_d}{n} n_i \rceil$.

Once we have the number of phrases of length i in the codeword, we can generate the phrase profile vector $\mathbf{v} = (v_1, v_2, \ldots, v_N)$ of length N. Here, N is the number of phrases in a codeword ($N = \sum_{i=d+1}^{k+1} n_i$) and v_n is the length of the n-th phrase.There are n_i values equal to i, for each i, in the phrase profile vector $\mathbf{v}$. Each phrase in the codeword will have a single nonzero symbol at the start of the phrase. For binary sequences, this symbol will always be a one, and the codewords will be generated simply by random permutations of the phrase profile vector. These permutations will provide $|U| = N!/(n_{d+1}! \ldots n_{k+1}!)$ codewords. Here, $|U|$ is the cardinality of the set of all distinct permutations of $\mathbf{v}$. In [14]a method for enumeration encoding of these phrases is given. For our analysis we do not need to worry about encoding all we require is that a valid codeword be used, therefore, we can simply use any permutation of $\mathbf{v}$ to generate a codeword.

For M-ary codes, the encoder can be viewed as two separate encoders, as in Figure 24. The binary encoder has been described above. The M-ary encoder consists of a length N

amplitude vector $\mathbf{a} = (a_1, a_2, \ldots, a_N)$, with $a_i \in \{1, 2, \ldots, M-1\}$. Each of these a_i will be the single nonzero symbol at the start of a phrase in the codeword. Then, mod-M precoding is applied to obtain an RLL $(d+1, k+1)$ sequence in the desired M-ary alphabet. After this, a symbol mapper maps the sequence to the bipolar alphabet used for recording $\mathscr{M} = \{\pm 1, \pm 3, \ldots, \pm(M-1)\}$.

The rate R of the code can be calculated as follows. We use $b_v = \log_2 |U|$ bits to select a phrase codeword from the set U. This phrase codeword determines the ordering of the phrases. We use $b_a = N \log_2 (M-1)$ bits to determine the amplitudes of the nonzero symbols in the phrases. Then, the rate is given by $R = (b_v + b_a)/n$, where n is the codeword length defined above.

4.4 Results

The simulations were performed using maximum-likelihood sequence detection with the Viterbi algorithm. The trellis is the one generated from the g_k channel. When RLL coding is used, we use a reduced state trellis, in which all the states and transitions that violate the d constraint of the RLL code are removed from the trellis. An example of this can be seen in Figure 25 for a $d = 2$ constrained code with channel taps $g_k = [1\ 0.3648\ 0.0248\ 0.0074]$ and binary antipodal signaling. There is no check for the k constraint. The trellis on the left is the full trellis for the g_k channel. The trellis on the right is the trellis for the same channel with the d constraint incorporated. Notice that incorporating the d constraint removes two states and eight transitions from the trellis. Figure 26 shows a block diagram of the simulation system. Here, the permutation code generator includes the (M, d, k) generator, the mod-M precoder and the mapper to symbols of the M-ary bipolar alphabet.

For all the simulations, a value of $\lambda = 780$ nm is assumed, with $NA = 0.5$. These values give us a minimum feature of $l_{MF} = \lambda/4NA = 390$ nm. If we assume $l_m = 600$ nm for uncoded transmission, then $l_m = 1.54 l_{MF}$. This would give a lineal density $D = 0.65$. Then, for $\beta = 0.2$, we find the channel coefficients using spectral factorization, which gives

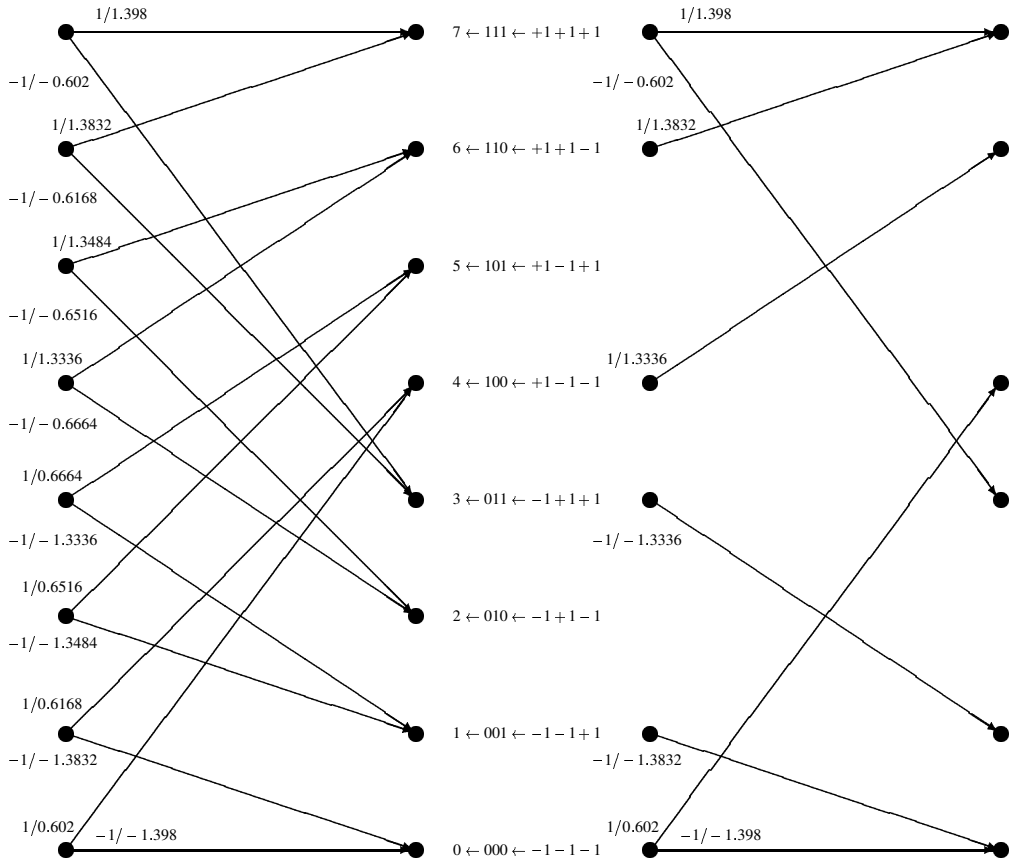


Figure 25: Trellis and reduced state transition trellis for Viterbi detection.

$g_k = [1\ 0.7135\ -0.0125\ 0.0053]$. For this channel, $A^2 = 10.08$ dB.

It has been shown (see Table 3) that as β decreases, the channel has less ISI. This will be evident in the plots of *SNR* vs. P_e, when we compare for different values of β. Another variable of interest is the laser wavelength λ. How does the channel change when λ increases? If l_m is kept constant at $l_m = 600$ nm, then an increase in λ will cause an increase in D, which according to Table 2 gives us a channel with more ISI.

4.4.1 Analysis of the RLL Coded Channel

It is of interest to see whether RLL coding is a good choice for writing signals in the M-ary optical recording channel. To evaluate this, we compare (M,d,k) RLL codes, for

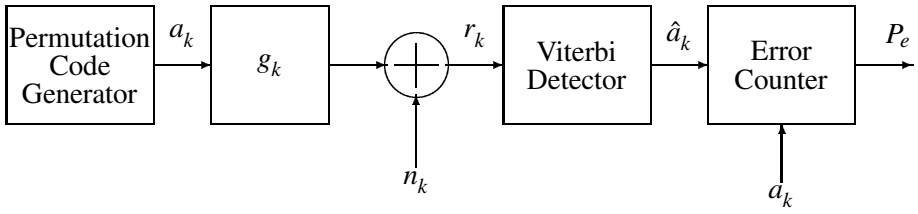


Figure 26: The simulation system.

$M = \{2,4,6,8\}$ and $d = 2,\ k = 10$, with an $M = 8$ uncoded signal and a real system employing trellis coded modulation (TCM) and one using Turbo codes. For both real ML CD-RW systems, the signals use $M = 8$. The codewords for the (M,d,k) RLL codes are generated according to the procedures described in Section 4.3. The code rates used are $R = \{0.52,\ 0.88,\ 1.08,\ 1.2\}$ for $M = \{2,4,6,8\}$, respectively. It is of interest to evaluate performance at a specified number of bits/(600 nm), since 600 nm is the size of the mark used in the real systems we are comparing against. Simulations were performed for densities of $\{1,\ 2,\ 2.5,\ 2.75,\ 3\}$ bits/(600 nm). The TCM code we are comparing against has a user bit density of 2.5 bits/(600 nm), while the density for the Turbo code is about 2.75 bits/(600 nm). The experimental data for these codes was obtained from [32]. The results are shown in Figure 27 for a channel with $\beta = 0.2$ and in Figure 28 for the channel with $\beta = 0.05$. We use two β values to see how performance is affected by this parameter.

Figure 27 shows that there is an advantage in using multilevel signaling over binary, as long as we code at a density greater than 1 bit/(600 nm). For a density of 1 bit/(600 nm), binary signaling outperforms multilevel. However, at 2 bit/(600 nm) we see that binary RLL is outperformed by all of the M-ary RLL codes. For this density, the marks for binary RLL are small. Therefore, there is considerable ISI in the channel, which degrades the performance. This density is the highest of the ones evaluated that can be achieved with binary signaling. The results also show that $M = 4$ is the best performing value of M, except for the density of 3 bits/(600 nm). At this density $M = 6$ outperforms $M = 4$ by about 0.5 dB. However, the performance gain is obtained at a high increase in detection

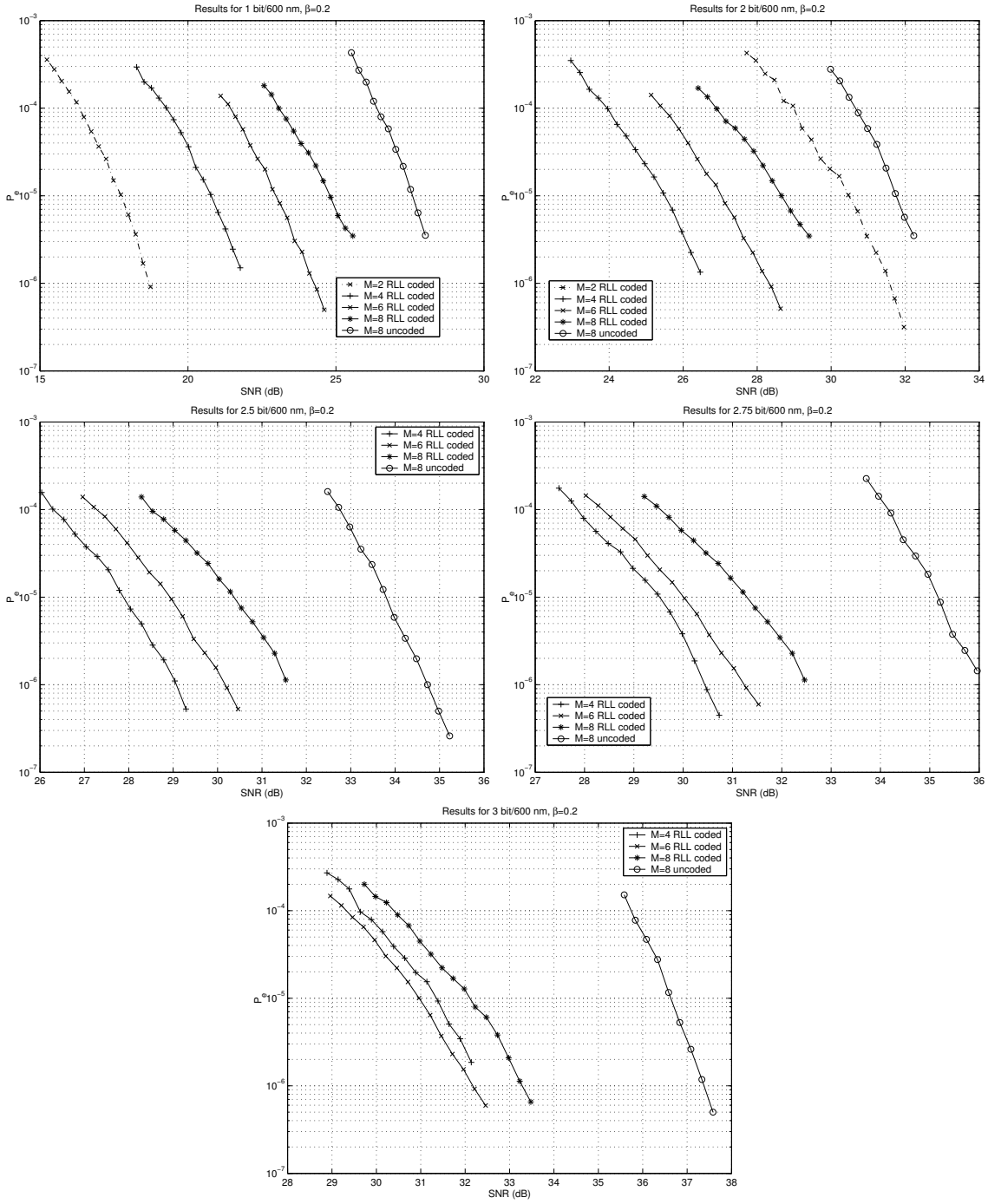


Figure 27: Results for $\beta = 0.2$ channel.

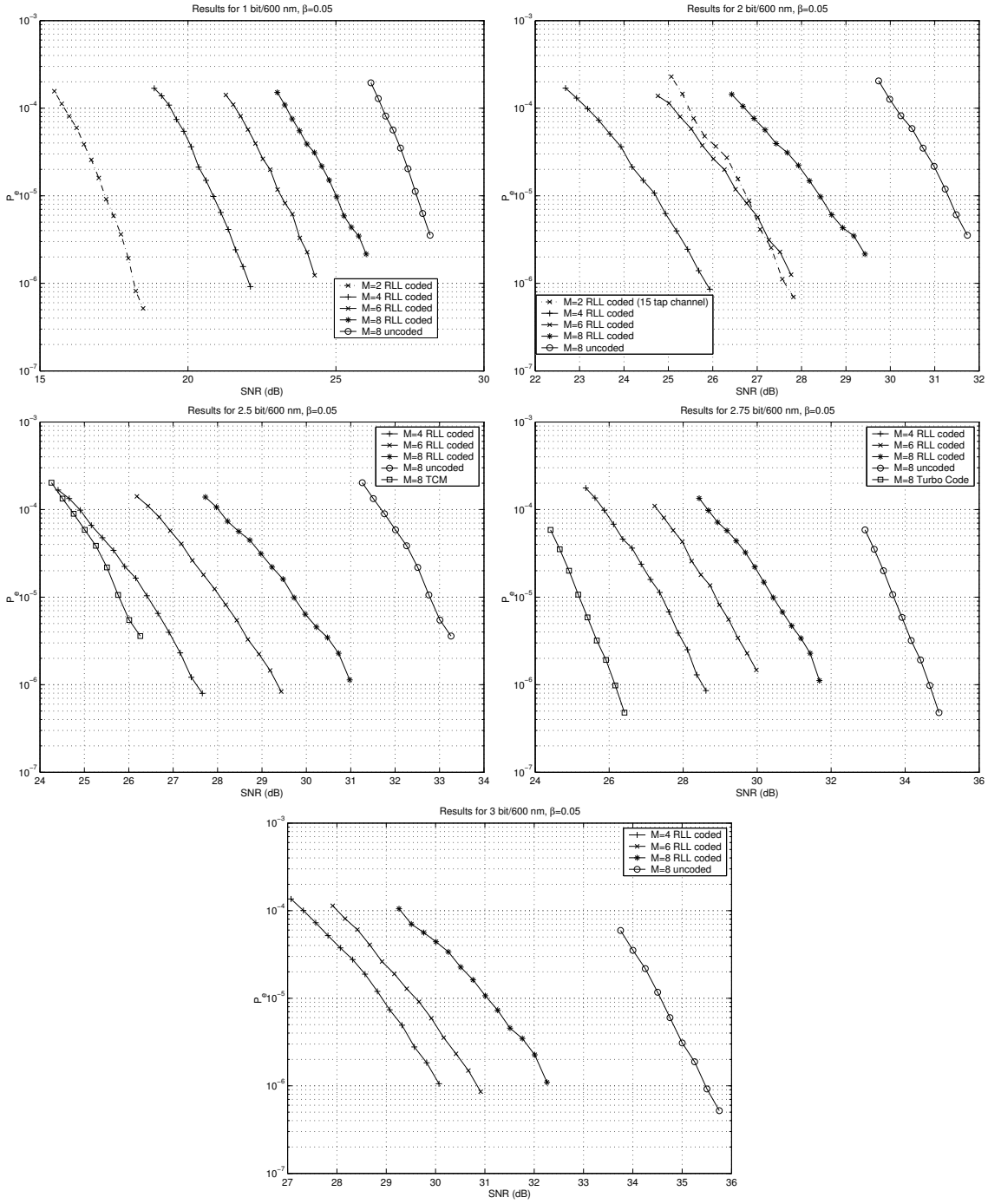
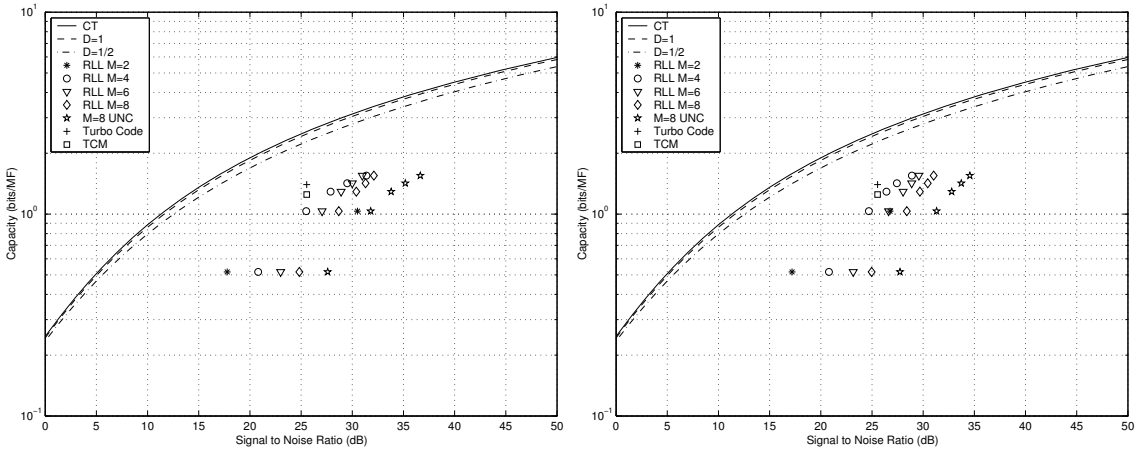


Figure 28: Results for $\beta = 0.05$ channel.

complexity. This, since using $M = 6$ adds a large number of states to the detection trellis. In particular, if we use a reduced state and transition trellis and the discrete-time equivalent channel has seven taps; the number of states increases from one hundred seventy one for $M = 4$ to six hundred and five for $M = 6$. Therefore, there is little to gain in using $M = 6$ instead of $M = 4$ for the bit densities evaluated.

Figure 28 shows the performance for a channel with $\beta = 0.05$. Performance for the $\beta = 0.05$ channel is better than that for the $\beta = 0.2$, for all user bit densities. This shows the increase in ISI as β increases. As expected, binary signaling performs better for density 1 bit/(600 nm). For a density of 2 bits/(600 nm) binary signaling is once again outperformed by M-ary signaling, although only by $M = 4$ and $M = 6$. The binary marks in this channel are larger than those for the $\beta = 0.2$ channel, due to the reduction in ISI. Therefore, performance degrades less with the increase in density, which accounts for the fact that binary signaling still outperforms $M = 8$ at this density. In the plots for 2.5 and 2.75 bits/(600 nm) in Figure 28 we show the performance of the ML CD-RW TCM and Turbo code systems. These fixed-length mark systems improve the performance of the uncoded system by about 6-9 dB. They also improve on the performance of the M-ary RLL coded systems. In particular, the TCM code system requires about 1.5 dB less of SNR to achieve $P_e = 10^{-5}$ than the $M = 4$ RLL coded system, which is the best performing M-ary RLL system at 2.5 bits/(600 nm). The Turbo code system requires about 5 dB less of SNR than the $M = 4$ RLL coded system to achieve the same error probability. For this channel $M = 4$ outperforms all other of the simulated M values for all of the densities evaluated. This is once again due to the reduction in ISI which allows for larger marks (than in the $\beta = 0.2$ channel), to achieve the same densities.



a) $\beta = 0.2$ b) $\beta = 0.05$

Figure 29: MLSD and theoretical capacity results.

4.4.2 Comparison with Theoretical Capacity Calculations

Figure 29 shows results both from the theoretical calculations and from the MLSD simulations. There are three lines in the plot, one corresponding to the capacity of the continuous-time channel model (solid line) and two corresponding to the capacity of the discrete-time channel model. The discrete-time plots correspond to two different values of the density D. For $D = 1$, the length of the mark written on the disc would be the same as that of a minimum feature. For $D = 1/2$, it is twice the length of a minimum feature. It can be seen that there is little to be gained by decreasing the length of the mark.

The points in the graph correspond to the SNR at which systems with a given rate achieve $P_e = 10^{-5}$ in the simulations. There are five clusters of points, one for each value of the user bit density $\{1,\ 2,\ 2.5,\ 2.75,\ 3\}$. For each user bit density, we have points for the different values of M. Notice that for $M = 2$, we only have points for the two lower densities. The reason for this is that we can't achieve the higher bit densities with binary signaling. To do so would require that our mark size be smaller than a minimum feature, something that is not possible. Once again, we can see that binary signaling performs better for 1 bit/(600 nm), but is outperformed by M-ary signaling for 2 bits/(600 nm). We also see that $M = 4$ is the best performing of the multilevel RLL codes. In these plots, we include

the results for the TCM and Turbo code systems. We can see that these coded systems perform about 3-5.5 dB better than the RLL coded systems for $\beta = 0.2$ and about 1-4.5 dB better for $\beta = 0.05$. However, there is still a considerable gap to capacity. The ideal coding and signal processing combination would reduce this gap to a few dB.

4.5 Conclusion

We have given a simplified discrete-time model for the optical recording channel. The model represents the channel as a discrete-time finite-impulse response with additive white Gaussian noise. Simulations with uncoded signals and RLL coded signals were performed using the discrete-time equivalent channel. The simulation results were compared with the theoretical capacity calculations, showing that multilevel signaling outperforms binary signaling for high user bit densities. The use of more advanced coding techniques should provide a larger improvement in performance than the one achieved with RLL coding, we will show this in the following chapters.

CHAPTER 5

CODED MODULATION I: COMBINED RLL/ECC CODING

The analysis in Chapter 4 shows the performance of a runlength-limited code over the model for the optical recording channel given in section 3.5. However, other than the RLL code, no error control code (ECC) is used. The fact that error control coding can improve the uncoded fixed-length mark system performance by about 6-9 dB leads us to believe that the overall performance of the RLL system can be improved if an ECC code is used in concatenation with the RLL code. With the addition of the ECC code, the system can be represented by Figure 1. Before specifying the ECC code, we need to describe the encoding and decoding of the RLL permutation codes. This is an issue that we did not address in previous chapters. However, it is not a trivial issue. Encoding high-efficiency permutation codes, such as the ones used in our previous analysis, requires a set of phrase codewords U with a large cardinality. The process of mapping bits to phrase codewords is critical and an efficient way to accomplish it needs to be found. In this chapter, we discuss efficient encoding/decoding algorithms for permutation codes. We also discuss the detection of permutation codes. In particular, what can we do if our detector outputs a codeword that is not in the set U?

5.1 Enumeration Encoding of (M,d,k) Permutation Codes

In Section 4.4.1, we assume that we can encode and decode high-efficiency (M,d,k) permutation codes. However, no method for encoding and decoding the codes is given. One particular method, based on an enumeration approach, can be found in [15]. Enumeration is an encoding/decoding approach first introduced for (d,∞) sequences in [26]. It was

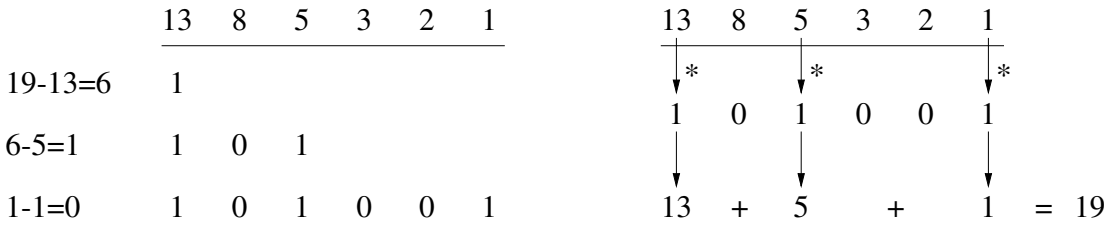


Figure 30: Example of encoding/decoding using enumeration.

first applied to (M,d,k) sequences in [44]. Enumeration approaches establish a one-to-one mapping between a set of integers and the set of codewords in the (M,d,k) code. Decoding is done by forming the weighted sum of the symbols in the received codeword, using weights that are a function of the constraints, thereby obtaining an integer that is then converted to its related codeword. Encoding is similar to decimal-to-binary conversion, except it uses the specified weights instead of the normal powers of two [38]. Figure 30 shows an example of encoding and decoding using enumeration. We see the relationship between the code and its weights. In this particular example the set of weights is $\{13,\ 8,\ 5,\ 2,\ 1\}$.

A permutation code enumeration encoding approach can be found for binary, $(2,d,k)$, codes. Let $\{n_d, n_{d+1}, \ldots, n_k\}$ denote the number of phrases of length n_i in any codeword, $N = \sum_{i=d}^{k} n_i$ the number of phrases in a codeword, and $h_i = N - \sum_{j<i} n_j,\ d \leq i \leq k$. Define R_i as

$$R_i = \begin{pmatrix} h_i \\ n_i \end{pmatrix}. \tag{27}$$

Then, the cardinality of the codeword set U can be expressed as

$$|U| = \prod_{i=0}^{K-1} R_i, \tag{28}$$

where $K = k - d + 1$ is the number of distinct phrases in a codeword, and R_i is defined above. Any number x in the range $[0, |U| - 1]$ can be uniquely represented by

$$x = b_o + \sum_{i=1}^{K-2} b_i Z_i, \text{ where } Z_i = \prod_{j=0}^{i-1} R_j,\ 1 \leq i \leq K-2. \tag{29}$$

The weights used for the encoding and decoding of the codewords are based on the R_i

and Z_i. The complete description of the encoding and decoding procedures can be found in [15].

The mapping restricts the cardinality of the (M,d,k) code. In particular, if we have only N_{int} bits to represent integers, our code has to have a cardinality $|U| < 2^{N_{int}}$. Otherwise, we will not be able to perform the one-to-one mapping. The high-efficiency codes used in Section 4.4.1 can't be enumeration encoded for the following two reasons:

- The cardinality $|U|$ of the phrase codeword set U is larger than $2^{N_{int}}$, the maximum cardinality we are allowed to represent.

- The codewords, as defined in Section 4.4.1, are pure (M,d,k) codes. They do not consist of two encoders as in Figure 24, where binary phrases and amplitudes are encoded separately. Instead, a single mapping from p bits to q symbols is used.

Therefore, we need to reduce the cardinality of the set U and encode phrases and amplitudes separately. To encode the phrases, we design an $M = 2$ code satisfying the d and k constraints. Enumeration encoding is applied to the $(2,d,k)$ code using the mixed-radix methods described in [15]. Our phrase set cardinality $|U|$ depends solely on the number of words in the $(2,d,k)$ code. The amplitudes are encoded separately. In our implementation the amplitude encoder is simply a symbol mapper. The (M,d,k) code is obtained by multiplying the ones in the phrases of the binary encoder by the appropriate amplitudes.

Typical CD-R/RW systems and DVD systems use the $d = 2$ and $k = 10$ constrained Eight to Fourteen Modulation (EFM) and EFM-Plus. In our simulations, we have changed the parameters to $d = 2$ and $k = 6$ to reduce even further the cardinality $|U|$ of the $(2,d,k)$ code space. This allows us to use enumeration encoding/decoding of the RLL sequences. Since our goal is to compare the performance of ECC and RLL concatenation with pure ECC fixed-length encoding, it does not matter whether or not we change the parameter k in our comparison. A larger value of k might provide further improvements, but we are not in a position to evaluate that with the resources used for the simulations. The drawback

from both changes is a reduction in the overall efficiency of the evaluated codes. The new codes that can be implemented with these changes achieve, at most, ninety percent efficiency. The efficiency of a code, defined in Section 2.3.1, is expressed as $E = R/C$. Therefore, reducing the cardinality of $|U|$ decreases the rate R of the code, which lowers the efficiency. However, enumeration encoding can not be implemented if the cardinality of $|U|$ is large.

5.2 Detection with Viterbi Detector

In the simulations for Section 4.4, we used a Viterbi detector in the uncoded case and a reduced state Viterbi detector for the RLL coded sequences. The difference between these two detectors is that the second one incorporates the d constraint into the trellis, effectively eliminating any state that does not satisfy this constraint (see Figure 25). The Viterbi detector is used to find the maximum-likelihood symbol sequence in the readout channel. This sequence can then be used to evaluate the symbol error probability for both, the uncoded and RLL coded case. The complexity of the Viterbi detector depends on two parameters: the memory of the channel (used to generate the trellis) and the alphabet size. If the size of the alphabet or the channel memory increase there is an exponential increase in the complexity of the algorithm. Incorporating the RLL constraint slightly reduces the complexity by decreasing the number of states and transitions in the trellis.

Figure 31 shows the block diagram for the implemented system, which contains an RLL permutation code. The system attempts to evaluate performance using the bit error probability as our performance measure. However, it is not possible to evaluate bit error probability using this system. This, due to the fact that it does not guarantee the detected RLL coded sequence will be a valid codeword of the RLL permutation code. In fact, symbol shifts in the detected sequence can cause the detected sequence to not satisfy the code constraints. When this happens, the permutation code decoding fails and an input bit sequence can not be found. For example, Figure 32 represents a codeword of a $(4,1,3)$

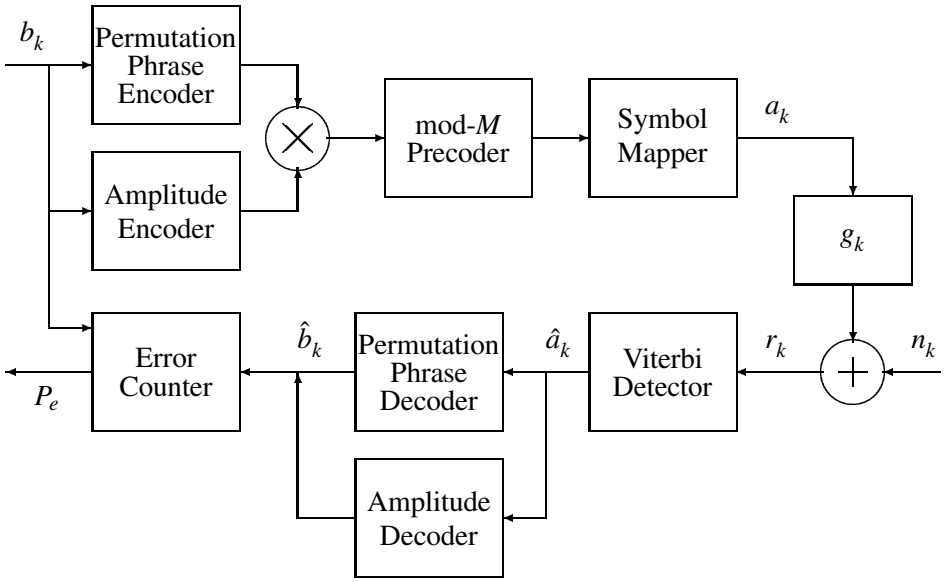


Figure 31: Complete permutation code system.

Recorded Word	−1	−1	3	3	3	−3	−3	3	3	3	3
Detected Word	−1	−1	3	3	3	−3	−3	−3	3	3	3

Figure 32: A symbol shift in an RLL permutation code.

code, precoded and mapped to channel symbols, and the detected sequence. This detected sequence satisfies the (d,k) constraints; however, it is not a codeword, due to the presence of a symbol shift. In this example, the permutation code has a phrase profile vector $v = (2,1,1)$. Therefore, the codeword has two phrases with a runlength of two, and one each with a runlength three and four. The detected word has one phrase with a runlength of two and three phrases with a runlength of three. In this case, the phrase decoder of the permutation code fails to decode. This is a major problem since symbol shifts are a common occurrence in optical recording [11].

To solve this problem, a detector that will guarantee that the detected RLL coded word

is a valid codeword is needed. There are three possible solutions to the problem encountered:

- One that incorporates all the permutation code constraints into the trellis used for detection. This is essentially a joint maximum likelihood (JML) detector. It uses its knowledge of the phrase profile vector of the code to guarantee that the detected word is a codeword satisfying the phrase profile and the (M,d,k) constraints.

- A post processing algorithm for the Viterbi detector. This algorithm would be called into action only when the detected codeword does not satisfy the phrase profile vector. It would then search for the most likely valid codeword based on the detected codeword and received symbols.

- Use another type of (M,d,k) code for which this problem does not exist or can be addressed easily.

The following sections explore each the first two ideas. The last possible solution is left for discussion in Section 5.3.

5.2.1 Joint Maximum-Likelihood Detector for Permutation Codes

We investigate the first approach, a detector that always selects a valid codeword, in our attempts to specify a complete encoding/decoding system for the ORC. To do so, we first need to specify the maximum-likelihood detector for the permutation code. We show that JML detection is not a viable option, due to its complexity. For simplicity, we consider the simple code defined above $M = 2$, $d = 1$, $k = 3$, and $v = (2,1,1)$. We then show how the JML detector changes with a larger value of M and less restrictive values of d and k. This procedure shows that the complexity of the JML detector becomes prohibitive as the difference between d and k, and the channel memory increase. Therefore, a sub optimum approach needs to be found.

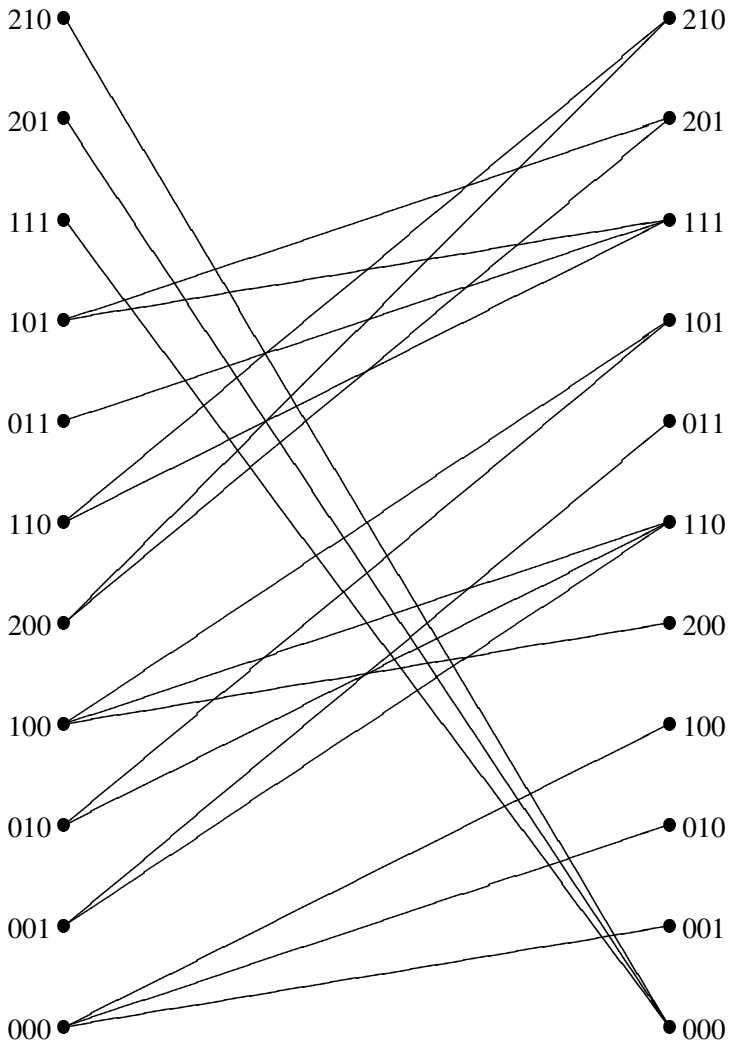


Figure 33: Trellis for a $\mathbf{v} = (2,1,1)$ permutation code.

Assuming a channel with no ISI and additive white Gaussian noise, Figure 33 represents the phase trellis for the $(2,1,3)$ permutation code with $\mathbf{v} = (2,1,1)$. Each state in the trellis is labeled by the number of phrases of each of the lengths already detected. Then, being in state 210 means that two phrases with a runlength of two and one phrase with a runlength of three have been detected. In this case, the only possible transition from that state is to state $211 = 000$. The detected phrase would have a runlength of four and complete a valid codeword. There are eleven states in the trellis, this number comes from $(v_1+1)*(v_2+1)*(v_3+1)-1 = 3*2*2-1 = 11$. The minus one in the formula comes

from the fact that state 211 is equivalent to state 000, or in general, state $v_d v_{d+1} \dots v_k$ is equivalent to state $00 \dots 0$. In both states, 211 and 000, detection of a new phrase is the detection of the first phrase in a new codeword.

We need four stages of the trellis in Figure 33 to detect a valid codeword in the code space, one stage for each phrase in the codeword. It can be shown that there will be twelve distinct paths through the trellis. This is equal to one path per codeword in the code space. In other words, the paths of distinct codewords only merge at the final state, so that the trellis is equivalent to performing a brute force search of the maximum-likelihood codeword over all the codewords in the permutation code. It can be shown that this extends to any permutation code. Thus, the ML detector for a permutation code is equivalent to a brute force search over the code space.

The trellis for joint maximum likelihood (JML) detection of a multilevel permutation codes is a product trellis of the trellis for the channel, which takes into account the amplitudes, and the trellis for the phrases in the codeword. The number of states in this trellis would be the multiplication of the number of states in each of the individual trellises. Then, for the $(2,1,3)$ code used above if the channel has ISI and consists of four taps, the JML trellis would be the product trellis of those in Figure 25 and Figure 33. The JML trellis would then have sixty six states for this simple code.

To show how complexity increases, let us evaluate an eighty eight percent efficient $(M,d,k) = (2,2,6)$ code of total length one hundred and twenty eight symbols. This code's phrase profile vector is given by $\mathbf{v} = (10,7,5,4,3)$. Then the number of states in the phrase profile trellis is three thousand nine hundred ninety nine. The total number of states in the joint trellis is close to twenty four thousand states. Performing this search for an (M,d,k) $= (8,2,6)$ is unreasonable, the computational complexity would be too high. Therefore, an alternative, suboptimal method of detection for permutation codes needs to be found. The next section discusses one possible implementation of such a method.

5.2.2 Post-Processing Algorithm for Viterbi Detection of Permutation Codes

Here, we discuss the implementation of a post-processing algorithm for Viterbi detection of a permutation code. Such an algorithm would have the following characteristics:

- It would be used only when the Viterbi detected codeword is not a valid codeword in the code space.
- It uses knowledge of the phrase profile vector, the received sequence, the Viterbi detected codeword, and the Viterbi trellis, to find the valid codeword.

There are three steps that are essential to the implementation of a post-processing algorithm:

- Establish a set of candidate valid codewords. The phrase profile vector tells us how many phrases of each length should have been detected. If there is only one symbol shift then there is one phrase length for which extra phrases exist and one phrase length for which we have less phrase than in the phrase profile vector. We can then construct valid codewords by modifying the boundaries of phrases of the length for which we found extra phrases. For example, for the detected sequence in Figure 32, the following are valid candidate codewords: $C_1 = \{-1,-1,3,3,-3,-3,-3,-3,3,3,3\}$, $C_2 = \{-1,-1,3,3,3,3,-3,-3,3,3,3\}$, $C_3 = \{-1,-1,3,3,3,-3,-3,-3,-3,3,3\}$ and $C_4 = \{-1,-1,3,3,3,-3,-3,3,3,3,3\}$.
- Evaluate the distance between the received sequence and the trellis path of each of these codewords. This is easily accomplished by filtering each candidate codeword with the channel model, used to generate the Viterbi detector trellis, and calculating its distance to the received sequence.
- Select as the detected codeword the one for which this distance is minimum.

The main problem with this algorithm, for the permutation code, is that the set of candidate valid codewords can be large. In our example, where the codeword consisted of four

phrases, one symbol shift gave rise to four candidate codewords. The number of candidate codewords per symbol shift can only increase as the number of phrases in the codeword increases. Even thought the post-processing algorithm is only used when the detected codeword does not satisfy the code constraints, its complexity can affect system performance.

5.2.3 Ninety Percent Efficient Permutation Codes

From our discussion on encoding and decoding of permutation codes we know that high-efficiency codes, such as those used in Chapter 4, can not be efficiently encoded. Therefore, the results in Figure 28 are the benchmark, but are not achievable. A more realistic goal would be to implement ninety percent efficient permutation codes. Figure 34 shows how the results change for this new efficiency. We consider only multilevel, as opposed to binary, signaling. For low user bit densities, 1 and 2 bits/600nm, $M = 3$ is the best performing RLL coded system. However, we can not achieve higher densities with this number of amplitude levels, since doing so would require marks smaller than a minimum feature. Therefore, for 2.5 and 2.75 bits/600nm, $M = 4$ is the best performing RLL coded system. Once again, $M = 4$ can not achieve higher densities, leaving $M = 6$ as the best performing system for 3 bits/600nm. Compared to the results in Figure 28, we see that there is a loss in performance of about 1-2 dB to achieve the same error probabilities. Also the achievable user densities for a low M value are reduced, for example $M = 4$ can achieve 3 bits/600nm if the efficiency is about 99%, but it can not achieve this density if the efficiency is at 90%.

In Figure 34 we show plots of SNR vs. P_e, where P_e is symbol error probability. We have already discussed the reasons why we can not evaluate the bit error probability of the system in Figure 31. Therefore, an alternative solution needs to be found. We need to be able to decode the user data that we encode and the system with permutation codes does not allow us to do so. The solution is to use a different type of RLL code and we discuss it in the following section.

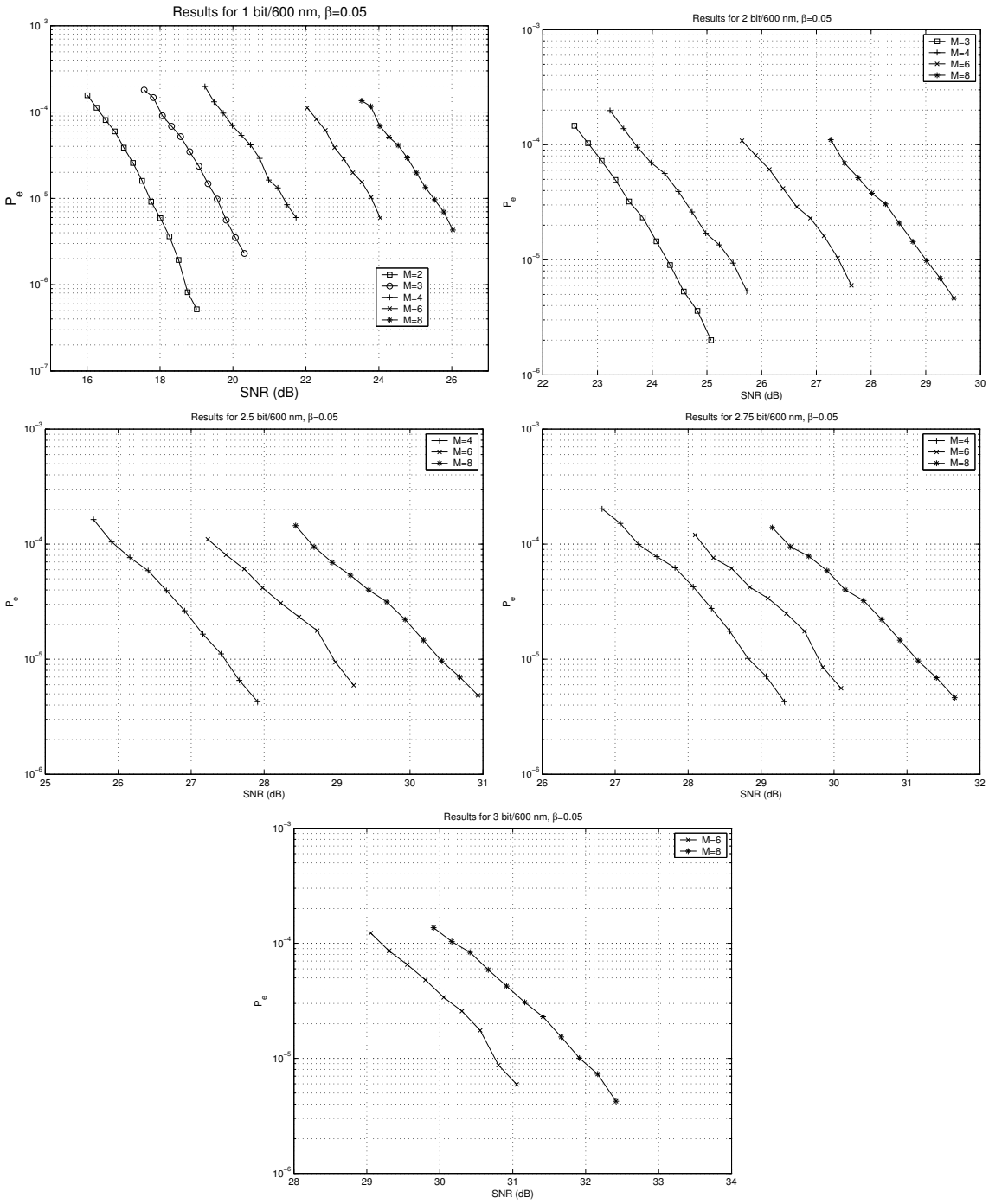


Figure 34: Results for ninety percent efficient permutation codes.

5.3 Combined ECC/RLL Coding

In Section 5.2 we observed that using a concatenation of an ECC with an RLL permutation code and Viterbi detection is not a good idea. An alternative needs to be found. We could use reverse concatenation, performing RLL encoding first and then using a systematic ECC code. This approach has been investigated for binary recording systems, see [1]. In general, it has been shown to perform better than the normal ECC/RLL concatenation. It's main drawback is the decoding latency, since most ECC codes used in the recording industry require a large block length for encoding/decoding. Also, an efficient algorithm for RLL coding the parity bits of the ECC needs to be found. Another approach is to use a code that combines RLL encoding with error-correction capabilities. This approach has the advantage of requiring only one encoder/decoder. Also no separate RLL encoding of parities is needed. The main advantage of using combined ECC/RLL codes is their ability to correct shift errors. This type of error is the most common error encountered in our channel model with the Viterbi detector. Therefore, a code that is designed to correct those errors is a highly desirable code.

This section deals with the design and implementation of a combined ECC/RLL code. We discuss two possible code constructions and give their advantages and disadvantages. Both constructions are based on Lee-Metric BCH codes. We then specify the reasons why we select one of those constructions. The specifics of the code implemented are also given. Plots of SNR vs. P_e for an encoding/decoding system are given. It was found that simple Viterbi detection is not enough to obtain adequate performance for large values of M. A specific solution to this problem is addressed in Section 5.5.

5.3.1 Lee-Metric BCH Codes

Lee-metric BCH Codes were first introduced in [27, 47]. Berlekamp [6] introduced the negacyclic codes, the most well-known Lee-metric codes, and provided an efficient decoding procedure for them. The first application of such codes to constrained channels was

given in [41], where they discuss the use of Lee-metric BCH codes to detect and/or correct synchronization errors in runlength-limited (d,k) channels. These errors are caused by the insertion and/or deletion of zero symbols in the runlength. In fact, an insertion error in one runlength followed by a deletion error in the next one is an example of what we have referred to as a symbol shift error.

The fact that these codes are designed to correct symbol shift errors has led us to believe that they are a good error correcting code for our specific application. At the same time that they correct errors, they can also satisfy the desired runlength constraint. Another advantage of these codes is that an efficient decoding algorithm has already been described. The binary RLL codewords obtained from the Lee-metric BCH code, according to the construction in [41], are of variable length. In [41] a construction of a binary RLL phrase encoder using Lee-metric BCH codes is given. The binary RLL codewords obtained from this construction are all of the same length. Therefore, a block encoder/decoder for this codes can be implemented.

5.3.1.1 The Lee Metric

The Lee distance $d_{\mathscr{L}}(x,y)$ between two elements $x, y \in GF(p)$ is the smallest absolute value of any integer congruent, modulo p, to the difference $x-y$. This metric is "circular" when applied to $GF(p)$, or put another way if all the elements in $GF(p)$ are arranged into a circle, the Lee distance between any two elements is the shortest of the arcs between the two elements. It is an alternative to the Hamming distance for non-binary signals. Therefore, given two codewords $\mathbf{u} = (u_1, u_2, \ldots, u_N)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_N)$, the Lee distance between the two codewords is given by

$$d_{\mathscr{L}}(\mathbf{u},\mathbf{v}) = \sum_{i=1}^{N} d_{\mathscr{L}}(u_i, v_i). \tag{30}$$

We see that it differs from the Hamming distance in that the Hamming distance only counts the number of coordinates that differ between the two codewords, not taking into account how much they differ by.

5.3.1.2 Code Construction

There are two possible code construction methods for combined ECC/RLL codes based on the Lee distance. The first method was introduced in [41]. We will call this method the Roth method. The code generated with this method is referred to as the $\mathscr{C}_q$ code. Encoding and decoding are simple and phrase length is obtained by a simple one-to-one mapping of elements from a field $GF(p)$ to a phrase length in the set of valid phrase lengths. This mapping is simple enough, the smallest field element represents the shortest phrase. The second smallest field element represents the second smallest phrase and so on. Some key characteristics of the code are the following:

- Simple encoding, multiplication by a generator matrix G to find the codeword in $GF(p)$. Then, simple one-to-one mapping to phrase lengths.
- Can correct insertion/deletions and symbol shift errors. The maximum Lee weight of errors that can be corrected is $r-1$, where $r \leq (p-1)/2$.
- Decoding algorithm based upon Euclid's algorithm, can correct errors of weight $r-1$ and detect all errors of weight r.
- Fixed number of phrases per codeword. However, the length of a codeword is not fixed. This is its main disadvantage.
- The size of the field p determines up to a constant the achievable code constraints. This, due to the one-to-one mapping between field elements and phrase lengths.

The main drawback of this code construction technique, the variable length of the codewords, is something that we would like to avoid, if possible. We are interested in using a code with fixed codeword length.

The code construction process described in [9] gives a code that is of fixed length. It uses as its starting point a code of the Roth type. We will call this underlying code the $\mathscr{C}_q$ code and the generated RLL code $\mathscr{L}$. Codewords from $\mathscr{C}_q$ are used to generate the RLL

codewords in $\mathscr{L}$. However, not all codewords from $\mathscr{C}_q$ generate codewords in $\mathscr{L}$, and some codewords in $\mathscr{C}_q$ may generate more than one codeword in $\mathscr{L}$. The key characteristics of codes designed by this process are:

- Fixed codeword length. All codewords in the code space $\mathscr{L}$ have the same codeword length N_2.
- Fixed number of phrases in a codeword. All codewords in the code $\mathscr{L}$ have the same number of phrases l.
- Can correct insertion/deletion and symbol shift errors. Once again, the number of errors that can be corrected is limited by q.
- There is not an efficient algorithm for encoding/decoding the codes. One could be found, perhaps by enumeration, but none have been given in the literature.
- The number of codewords in $\mathscr{L}$ can be large. Thus, even if an efficient enumeration algorithm to encode/decode the codes is found, problems such as those with enumeration of permutation codes can still arise.
- Most code spaces $\mathscr{L}$ that can be designed are characterized by low efficiencies. This makes them difficult to use in our system, since they limit the achievable user bit densities.

This code construction process is then ill-suited for our application. Its one desirable quality, the fixed codeword length, is outweighed by a set of non-desirable characteristics. Therefore, it is better to use the Roth code construction. Section 5.3.2 describes the phrase encoder used in our simulations.

5.3.2 Lee-Metric Phrase Encoder

The code $\mathscr{C}_q$ designed using the Lee-metric BCH approach is a $C(48,3,\alpha)$ code over $GF(7)$. The vector α is a vector with every single field element in $GF(49 = 7^2)$, represented as a power of a generating element and in ascending order. We can express it using MATLAB notation as $\alpha = [0:47]$. The parity-check matrix for this code is of the form

$$H(n,r,\alpha;p) \stackrel{\text{def}}{=} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \cdots & \alpha_n^{r-1} \end{bmatrix}. \tag{31}$$

Since $r = 3$ the matrix has only three rows of elements in $GF(49)$. Expressed as elements of $GF(7)$, the parity-check matrix consists of five rows. Finding the generator matrix is simple. First, the parity check matrix is expressed in systematic form $H(n,r,\alpha;p) = \begin{bmatrix} -P^T & I \end{bmatrix}$. Then, the generator matrix can be found as $G(n,r,\alpha;p) = [I \quad P]$. For our specific example the I matrix in the parity-check matrix is a five-by-five identity matrix, while $-P^T$ is a five-by-forty-three matrix, with elements of $GF(7)$.

5.3.2.1 Encoding Procedure

If we are interested in calculating the bit error rate we need to perform the complete encoding from bits to the RLL codeword. To do so we use the following algorithm:

1. Map a set of p bits into $q = 43$ field elements from $GF(7)$. This process forms the message word $\mathbf{m}$.

2. Multiply the message word $\mathbf{m}$ by the generator matrix to obtain the encoded word $\mathbf{c} = \mathbf{m}G = [c_1\ c_2\ \cdots\ c_n]$.

3. Construct the differentially precoded word $\mathbf{d} = [d_1\ d_2\ \cdots\ d_n]$, where $d_1 = c_1$ and $d_j = c_j - c_{j-1}$ for $2 \leq j \leq n$, with the subtraction taken modulo p.

4. Convert each field element in the precoded word **d** into the specific (d,k) constrained phrase. To do so, use the following conversion rule: write a '1' symbol and follow it by $d+d_i$ '0'.

The precoding step guarantees that we can correct up to $r-1$ bit shift errors. If we are only interested in computing symbol error probability, we can substitute step one in the above process by a random generator of message words **m**.

5.3.2.2 Decoding Procedure

For decoding we use the algorithm described in [41]. However, we must first remove the effects of precoding. We can reconstruct the original codeword **c** by the operation

$$c_j = \sum_{l=1}^{j} d_l. \tag{32}$$

If a shift error occurs at the boundary between runs j and $j+1$ in **d**, it is converted into an insertion/deletion in run j of **c**. For this to work we need d_1 to be correct. This can always be guaranteed if the code contains the all-one word and all its multiples. It is simple to check that our code does satisfy this.

5.3.2.3 Encoding Rate

We can estimate the encoding rate of our code. To do so, we calculate the number of bits that can be encoded into one codeword. We use one hundred and twenty bits to select one of the forty three symbol message words **m**. The number of symbols we encode using these bits is variable. However we can estimate the average codeword length with the formula

$$\bar{n}_{ROTH} = K\left(\frac{k+d}{2}+1\right) + (N-K)\left[1+\left(d+\frac{q-1}{2}\right)\right], \tag{33}$$

where $K=43$ is the number of information phrases, $N=48$ is the number of phrases in the code, $q=7$ is the size of the field, and d, k are the constraints of the code. Then in our case $\bar{n}_{ROTH}=288$ symbols. Then, the average encoding rate is $R_{ROTH}=0.42$ bits/symbol for the binary code.

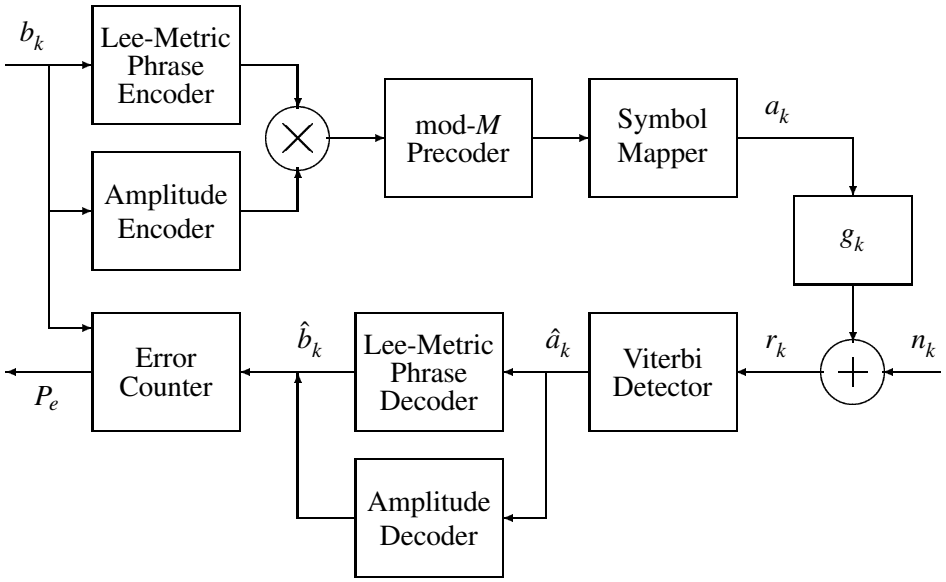


Figure 35: Complete Lee-metric BCH code system.

5.4 Results for ECC/RLL Combined Code

Figure 35 shows a complete diagram of the system, which replaces the binary permutation phrase encoder in Figure 31 with a Lee-metric combined ECC/RLL code. The code is designed to correct symbol transition shift errors, since these type of errors are the dominant source of errors in the optical recording channel. An example of this type of error was given in Figure 32. There are numerous examples of symbol transition shift error correcting codes in the literature [7, 9, 11, 17, 22, 41, 53]. Some of these codes can also correct substitution errors, one in which a '0' is replaced by a '1' or a '1' by a '0' in the RLL code, such as, those described in [9, 41]. These errors add/remove zeros to one of the runlengths of the RLL code. In particular a transition shift error can be interpreted as an insertion error in one phrase followed by a deletion error in the next phrase. The code used is the one specified in Section 5.3.2. Performance is evaluated for this code at user bit densities of $\{1,\ 2,\ 2.5,\ 2.75,\ 3\}$ bits/600nm.

Figure 36 shows plots of SNR vs. P_e for the Lee-Metric BCH code system of Figure 35. We can see that for low user bit densities, for example one and two bits/600nm, the system

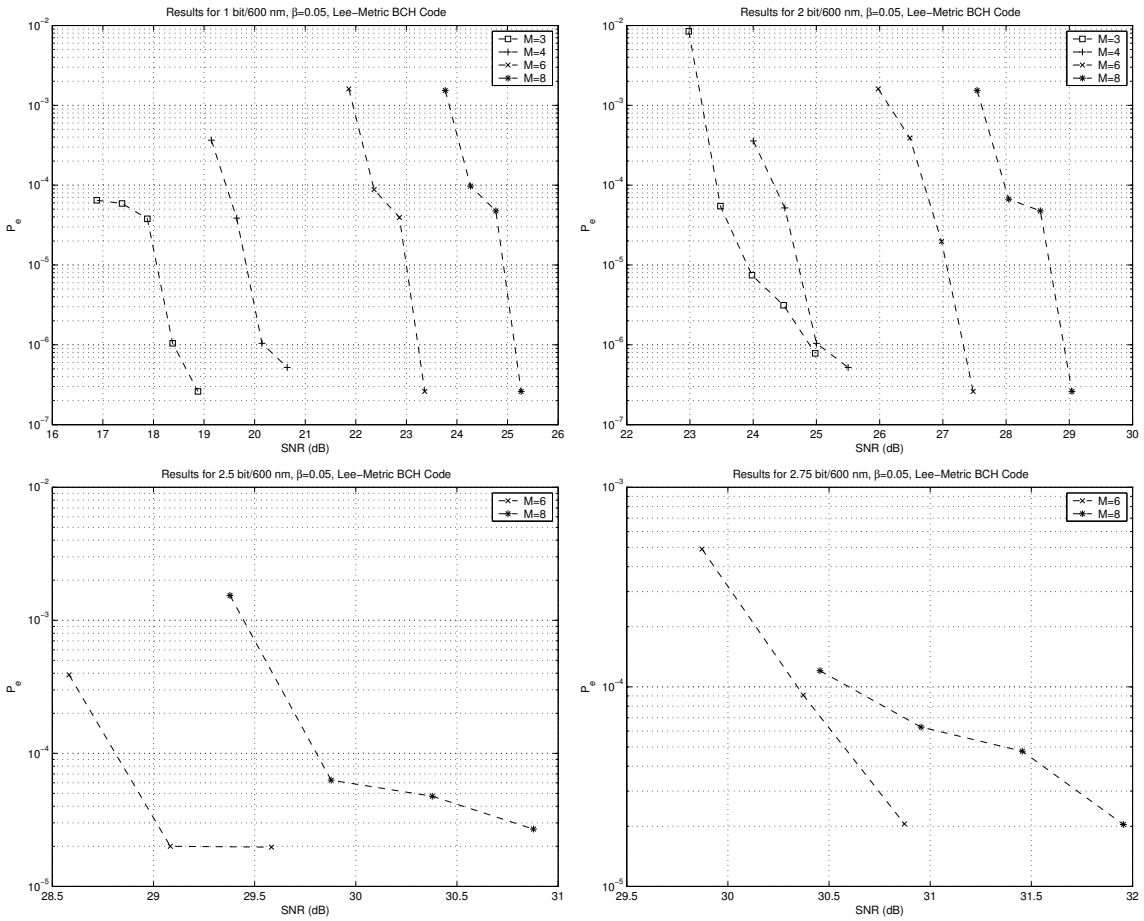


Figure 36: Results for system in Figure 35.

performs reasonably well. For these densities the codes with $M = 3$ are the best performing codes. Densities at or above 2.5 bits/600nm require that we signal with at least $M = 6$ levels. The performance for these cases is severely degraded and exhibits the presence of an error floor.

An analysis of the results for 2.5 and 2.75 bits/600nm tells us that most of the errors originate from Viterbi detection of a word that does not satisfy the code constraints. This happens since the Viterbi detector does not ensure that the detected codeword must have forty-eight phrases. If a word with forty-nine or forty-seven phrases is detected, one that is not a valid codeword in the Lee-Metric BCH code, then multiple errors exist and an error floor occurs. This problem can be solved by including a Viterbi post-processing

algorithm in the system in Figure 35. This post-processing algorithm has to be simple and guarantee that its output codeword will have exactly forty-eight phrases, satisfying the code constraints. It should only be used when Viterbi detection has produced a codeword with more, or less, phrases than those in all codewords in the code space. This algorithm is introduced in Section 5.5.

Another source of errors is a phrase that exceeds the k constraint. Let us assume that a phrase in the detected codeword has length $k+n$, for some $n \in [1, p-1]$. Then, due to the mod p operation of the finite field the BCH code belongs to, that phrase will be interpreted as a phrase of length n. This will in general give rise to multiple errors in the decoding process. Therefore, the post-processing algorithm needs to eliminate phrases exceeding the k constraint, by combining the n extra symbols into the adjacent phrases.

5.5 Post-processing Algorithm for Viterbi Detector

In Section 5.3, we introduced a combined ECC/RLL phrase encoder for the system in Figure 35. This code has good performance when the user bit density is low. For high user bit densities, those above 2 bits/600nm, the system performance exhibits an error floor. This error floor was caused by the erroneous detection of codewords, such that, the total number of phrases in the detected codeword was more, or less, than the actual number of phrases per codeword. In this section we solve this detection problem.

The approach taken is to add a post-processing block in detection. This block follows the Viterbi detector and is located before the decoding block corresponding to the phrase encoder. Its sole purpose is to guarantee that the detected codeword has the correct number of phrases. For our ECC/RLL code that number is forty-eight phrases. The post-processing block is used only when this is not the number of phrases in the detected codeword. If it is, we simply by-pass post-processing. The post-processing algorithm must be simple and general. Ideally, it should correct all codewords with an inaccurate number of phrases. The use of this algorithm should improve system performance on the SNR vs. P_e plots by

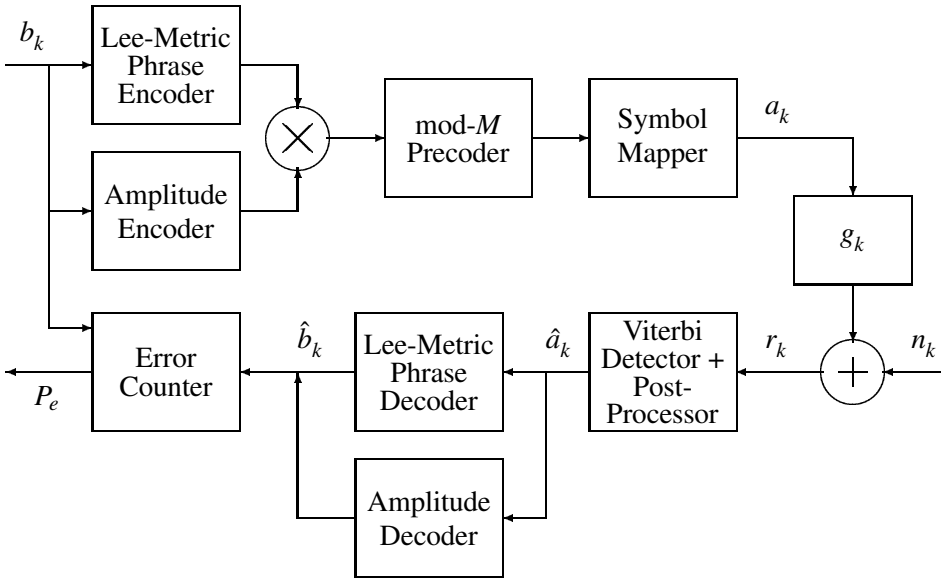


Figure 37: Complete Lee-Metric BCH code system with post-processing.

reducing the SNR value necessary to achieve a specific P_e.

5.5.1 New System Diagram

Figure 37 shows a diagram of the system with Viterbi plus post-processing. This is essentially the same system as that in Figure 35, with the Viterbi detector replaced by a Viterbi detector plus post-processing block. We have represented it in this way for visualization purposes. The post-processing block is actually an independent block that may or may not be used. In essence one of the following can happen:

1. The Viterbi detected codeword has the right number of phrases and no phrase exceeds the k constraint. In this case no post-processing is needed and we can proceed directly with decoding of the Lee-Metric BCH codeword.

2. The Viterbi detected codeword has the right number of phrases, but one or more phrases exceed the k constraint. Therefore, post-processing to decrease the length of phrases exceeding the k constraint is needed.

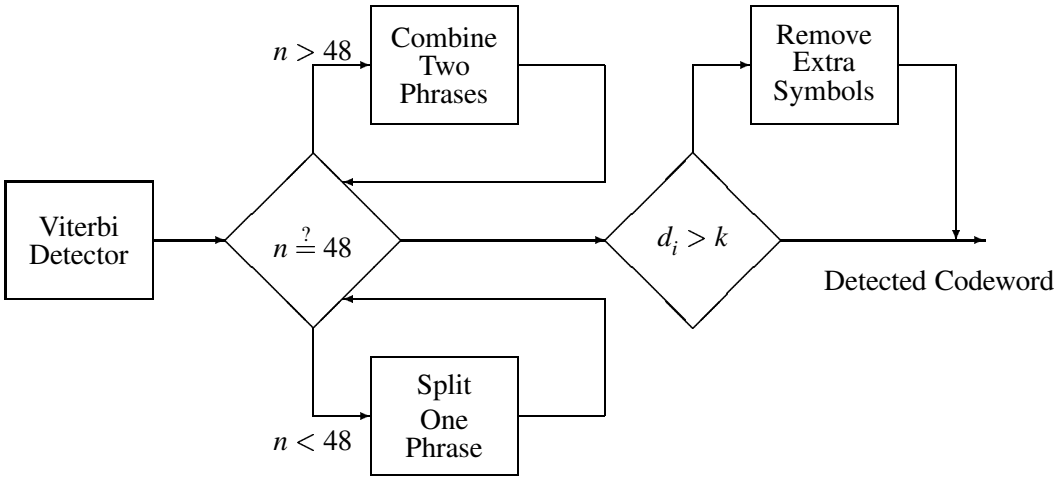


Figure 38: Diagram of post-processing algorithm.

3. The Viterbi detected codeword has more phrases than the correct codeword. In this case post-processing should be used to combine short phrases. Each time the process is used one phrase is removed. The process should be repeated until the detected codeword has the correct number of phrases.

4. The Viterbi detected codeword has fewer phrases than the correct codeword. Post-processing should be used to split one long phrase. Each time the process is used one phrase is added. The process should be repeated until the detected codeword has the correct number of phrases.

Figure 38 shows a diagram of the points listed above. There are three distinct post-processing blocks, two deal with the number of phrases and one deals with the length of the phrases. Of the blocks dealing with the number of phrases, one removes phrases by combining short phrases and one the other one adds phrases by splitting long phrases. The specific implementation details of each one of the post-processing blocks are described in Sections 5.5.2.1- 5.5.2.3.

5.5.2 The Post-processing Algorithm

As shown in Figure 38, the post-processing algorithm is divided into two different algorithms. One of the algorithms removes phrases, by combining two short phrases. The other adds phrases, by splitting one long phrase into two short phrases. However, both algorithms have some common elements, a routine that finds the positions in the detected codeword that are most likely to be in error and the criterion used to stop searching for a corrected codeword.

To find these positions we follow the path, through the trellis, of the detected codeword. The distance $d(c_i, r_i)$, to the received sequence, is calculated for each symbol. A cumulative distance measure is also kept at each symbol. This cumulative distance can be expressed as

$$d_{cum}(i) = \sum_{j=0}^{i} d(c_j, r_j) \tag{34}$$

The variation of the cumulative distance over the minimum mark length $d+1$ is given by $d_{d+1}(i) = d_{cum}(i) - d_{cum}(i-d-1)$. Then, we use the position at which this distance measure is maximum to find the positions most likely in error.

To determine whether a valid corrected codeword has been found, we first calculate the total cumulative distance of the detected codeword, the one with the wrong number of phrases. Assume the number of symbols in the codeword is l, then we call this distance $d_{cum}(l)$. Then, phrase combination or splitting is performed in and around the positions most likely in error in this codeword. The total cumulative distance $\hat{d}_{cum}(l)$ of the new codeword is calculated. These two distances are compared, if they are close enough, it is declared that a valid corrected codeword has been found. We have assumed that the distances are close if they are within ten percent of each other. If the corrected codeword is not valid, then we keep the one with the lowest total cumulative distance, find the next highest value in $d_{d+1}(i)$ and repeat the process until a valid corrected codeword is found or a maximum number of positions is evaluated. If this last condition is reached, the codeword we output is the corrected codeword with the lowest total cumulative distance.

5.5.2.1 Removing Phrases

To remove a phrase, we combine two or more short phrases. It is required that the combined phrases be short, so that the new phrase does not exceed the maximum allowable phrase length. Therefore, once we are given the most likely positions to be in error we need to find if it is possible to combine phrases close to those positions and still satisfy the code constraints. If it is possible, then we do so and evaluate the total cumulative distance. We combine phrases in one of the following ways:

- Forward Combination: The combined phrase will consist of the symbol of the first of the two phrases being combined.
- Backward Combination: The combined phrase will consist of the symbol of the second of the two phrases being combined.
- Split Combination: We eliminate one small phrase by combining it with the phrases next to it. The first few symbols take on the value of the symbols from its preceding phrase. The remaining symbols take on the value of the symbols from the next phrase.

When performing these combinations we need to be certain that a phrase longer than the maximum allowable length is not being formed. To do so, in forward combinations we need to check that the symbol value of the phrase that follows our second phrase is not the same symbol value as that of the first phrase. If this is the case, then we cannot perform the forward combination. A similar check needs to be evaluated when performing backward combination. The main difference is now we check the phrase immediately before the first phrase being combined. The one exception to this rule is when the detected codeword has two or more extra phrases. In this case we allow a combination of three phrases if the combined phrase does not exceed the k constraint. This combination process may give us more than one codeword. If this happens, the codeword with the smallest total cumulative distance is the one returned as our candidate codeword.

5.5.2.2 *Adding Phrases*

The process for adding phrases involves splitting one phrase into two distinct phrases. This process is easy if one of the phrases in the detected codeword is longer than the maximum allowable length. This can happen since our trellis has no check for the k constraint. If we have such a phrase in our codeword, we know that we need to split the phrase. Phrase splitting can be accomplished using the following system:

- Check whether the current phrase symbol is one of the boundary symbols in the alphabet.
- If the current symbol is not a boundary symbol we can split the phrase with one of four approaches:
  - Forward Lower Value: In this case we split the phrase into two, assigning a lower symbol value to the first phrase and leaving the second phrase unchanged.
  - Forward Higher Value: The first phrase is assigned a higher symbol value and the second phrase is left unchanged.
  - Backward Lower Value: The first phrase is left unchanged and a lower symbol value is assigned to the second phrase.
  - Backward Higher Value: The first phrase is left unchanged and a higher symbol value is assigned to the second phrase.
- If the current symbol is a boundary symbol, then only the lower or higher approach can be used, depending on which boundary symbol it is.
- If no valid candidate codeword is found then we check whether we can create a new phrase in the boundary between two phrases. To do this we need to satisfy:
  - The symbol values between the two phrases have to differ by exactly two positions in the symbol alphabet. For example, in a 4-ary alphabet the symbols -3

and 1 differ by two positions.

- Each of the two phrases has a length greater than $d+1$ and a phrase of length $d+1$ can be created by combining symbols from these two phrases. As an example, if both phrases have length $d+2$ we can't create a phrase of length $d+1$ if $d>1$ since we only have one extra symbol from each of the two phrases.

We still assume that a check of the symbol values of the phrase immediately before and immediately after the phrase being split has been performed. From the possible ways to split a phrase we have to remove those that cause a conflict with these two phrases. For example, assume the phrase immediately before the phrase we are splitting has a symbol value of -3 and our phrase has a symbol value of -1. In this case, we can't use the forward lower value approach to split our phrase. This causes the first new phrase to recombine with the previous phrase and therefore no phrases would be added.

If there are no phrases that exceed the maximum allowable phrase length, then we split those phrases that are long enough to guarantee two phrases of at least the minimum phrase length. The process is still guided by the location of the symbols most likely to be in error. Notice that when performing phrase splitting there are more codewords to evaluate. This, since a phrase of length ten can be split into two of length five, one of length six and one of length four, or one of length seven and one of length three. For the last two cases we can place the short phrase first or we can place it second which gives us two other cases to evaluate. Therefore, we need to be careful and make sure that we evaluate all possible cases.

5.5.2.3 Phrases Exceeding the k Constraint

The trellis used in the Viterbi detector includes the code's d constraint. However, it does not include a check for the k constraint. Therefore, we can have detected codewords with phrases whose length exceeds the k constraint. This type of phrases causes problems in the decoding of the Lee-metric BCH code. Hence, we need to implement an algorithm

Table 5: Error Distribution for $M = 5$, 1 bit/600nm.

SNR	P_e	Amp.Err	Symbols	Shift Err	Symbols	Total Errors
16	0.000147486	6	35	59	69	104
16.25	9.39078e-05	8	43	51	60	103
16.5	3.27966e-05	4	18	69	82	100
16.75	2.07756e-05	2	10	81	92	102
17	1.45351e-05	2	12	84	91	103
17.25	8.50437e-06	2	12	79	86	98
17.5	4.07863e-06	0	0	42	47	47
17.75	2.5166e-06	0	0	28	29	29
18	1.30169e-06	0	0	15	15	15

that reassigns symbols to phrases so that no phrase length exceeds the k constraint. This algorithm is part of the post-processing stage in our detector. It performs the following tasks:

- Identifies phrases that exceed the k constraint.
- Finds the possible locations where the exceeding symbols can be moved to, by analyzing the phrase length of the phrases immediately before and immediately after the one exceeding the k constraint.
- Reassigns the extra symbols and calculates the distance of the new codeword. It selects as a valid candidate codeword the one with the lowest distance, since there may be two or more ways to reassign the symbols.

When performed, this process is the last step in the post-processing algorithm. Therefore, the output of this process is the new detected codeword which will be decoded by the Lee-metric BCH decoding algorithm.

5.6 System Performance with Viterbi Post-processing

Table 5 provides the breakdown of the errors for the combination of Viterbi detector and post-processing for $M = 5$ and a user bit density of 1 bit/600nm. The simulations stopped

Table 6: Error Distribution for $M = 5$, 2.5 bit/600nm.

SNR	P_e	Amp.Err	Symbols	Shift Err	Symbols	Total Errors
16.5	9.88674e-05	8	52	43	54	106
16.75	4.41031e-05	6	35	56	67	102
17	2.91408e-05	4	23	70	78	101
17.25	1.33865e-05	0	0	90	101	101
17.5	8.97816e-06	7	32	63	70	102
17.75	6.76878e-06	7	32	42	46	78
18.25	5.20676e-07	0	0	6	6	6

when at least one hundred symbol errors were found or after simulating forty thousand codewords, whichever came first. We can see that for high SNR values most of the errors are still caused by symbol shifts in the corrected codeword. These errors are those that the combined ECC/RLL code cannot correct. They are present when there is a detected word with an error pattern of Lee-weight larger than the error correction capabilities of the code. As the user bit density increases, see Table 6, the percentage of errors due to symbol shifts decreases. There are more errors caused by amplitude errors as the bit density increases. An increase in the value of M also increases the percentage of errors due to amplitude errors. Then, for $M = 8$ the percentage of amplitude errors is highest when the user bit density is highest. This leads us to believe that performance can still be improved if we use coding on the amplitude encoder, instead of a simple symbol mapper. Ideally, this amplitude coding should be applied when the SNR is low, and for large user densities and M values. The use of error-correcting codes in the amplitude encoder will be studied in Chapter 6.

Figure 39 shows a comparison of the results with post-processing and those without post-processing for $M = 8$ and a density of 2.5 bits/600nm. We see that there is a dramatic improvement in performance brought about by the use of the post-processing algorithms. In particular, there is about a 2 dB difference in performance at an error probability of about 7×10^{-4}. If we compare the results with post-processing in Figure 40 to those without it (Figure 36), we can see the improvement brought about by the post-processing. This improvement is more noticeable for user densities above 2 bits/600nm. These densities had

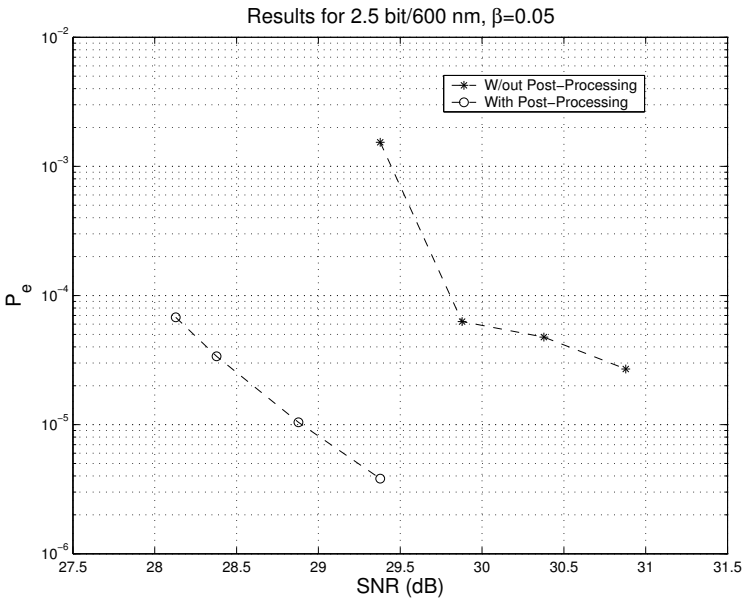


Figure 39: Comparison of results with and without post-processing.

performance curves exhibiting error floors when post-processing was not used.

In the plots for 1 bit/600nm and 2 bits/600nm, we have included the case of binary signaling. We can see that binary signaling has the advantage over M-ary signaling when the user bit density is low. In this particular case binary signaling outperforms M-ary signaling for 1 bit/600nm. At a density of 1.5 bit/600nm 3-ary signaling outperforms binary signaling. This density is the highest achievable bit density with binary signaling, assuming a Lee-Metric BCH binary encoder is used to encode the RLL phrases. For 1.5 and 2 bits/600nm we observe that the $M = 3$ RLL code is the best performing code. The performance of $M = 3$ and $M = 4$ at 2 bits/600nm is similar. However, there is an advantage in using $M = 3$ since the detection trellis has less states and detection is less computationally intensive. As the user bit density increases we need to increase the number of amplitude levels in order to be able to achieve the density. A density of 2.5 bits/600nm requires that we signal with at least $M = 5$ levels, which is also the best performing code for this density. For 2.75 bits/600nm we need to signal with at least $M = 6$ levels, which is also the best performing code for the density. In this case, the performance of $M = 8$ is very close to

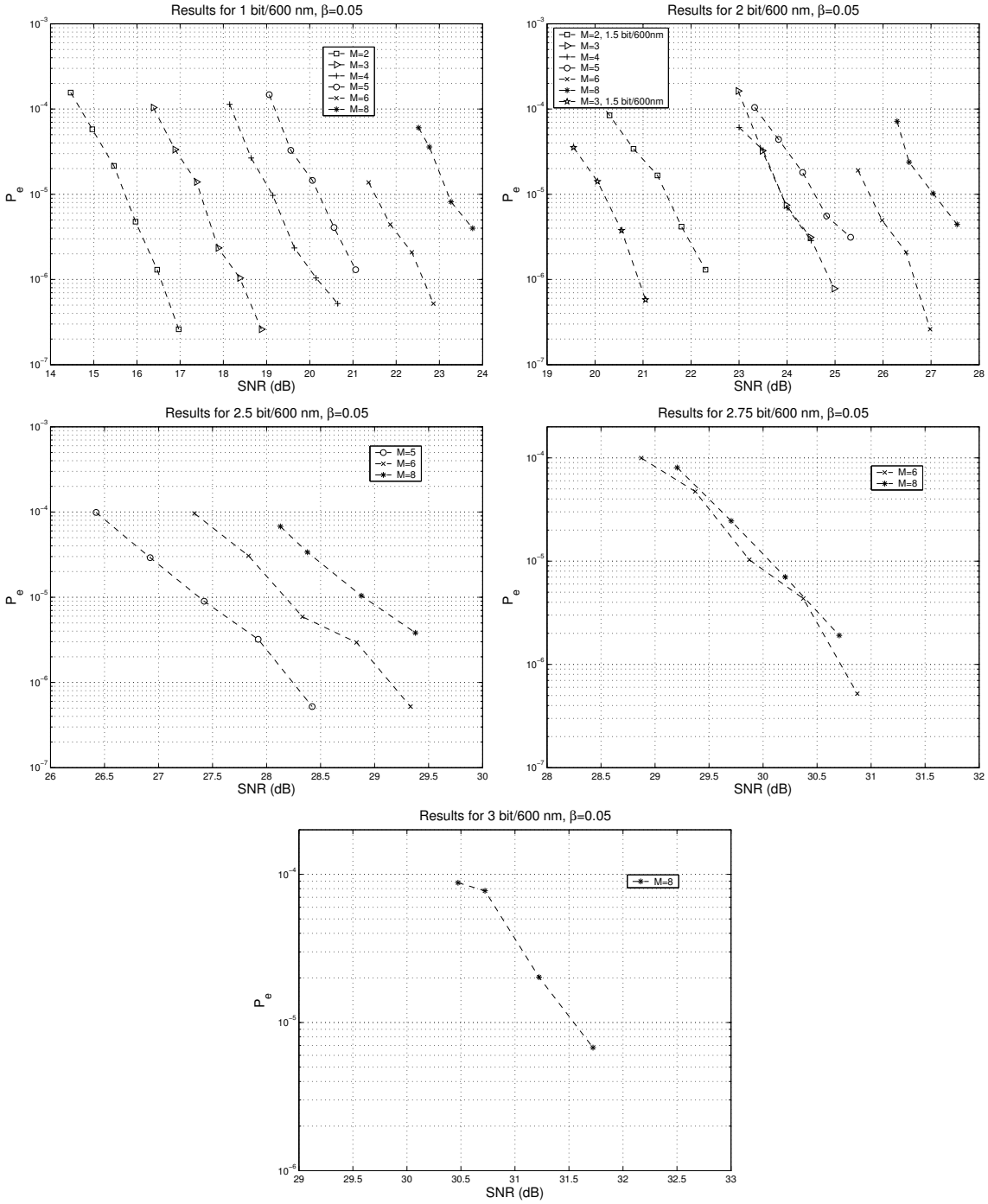


Figure 40: Results for system with Viterbi plus post-processing.

that of $M = 6$. However, $M = 6$ is preferred to the lower complexity of detection which arises from a trellis with less states. For 3 bits/600nm we need to signal with at least $M = 8$ levels.

5.7 Conclusion

In this Chapter, we have implemented a complete coded modulation system. We showed that using permutation codes in combination with Viterbi detector fails. This, due to the fact that the Viterbi detector does not guarantee that the detected word is a codeword in the code space. Two solutions to this problem were investigated: joint maximum-likelihood detection of phrases and amplitudes, and a post-processing algorithm for the Viterbi detector. For this type of code both solutions suffered from a high complexity and were not implemented.

A solution using a combined error-correcting/runlength-limited code (ECC/RLL) was proposed. The main advantages of using this type of code are the fact that only one encoder/decoder is required and the code's ability to correct symbol shift errors, which are the main source of errors in the Viterbi detector. This errors were also the main cause behind the failure of the permutation code. Two code construction methods were studied and a construction based on a Lee-Metric BCH code was used. The main drawback of the code is that it does not have a fixed codeword length. Another drawback is that the error correction capabilities of the code are linked to the code constraints (d, k). Therefore, if we want to increase the Lee-weight of correctable errors we have to relax the k constraint. Simulations showed that this code worked well for low user bit densities. However, it exhibited an error floor when high user bit densities were used.

Finally, we introduced a novel post-processing algorithm for Viterbi detection. The purpose of the algorithm is to guarantee that the number of phrases in the detected codeword matches the number of phrases in the Lee-metric BCH code and that no phrase length violates the k constraint. This eliminates the error floor exhibited by the simulations when

no post-processing was present. The post-processing algorithm returns the most-likely corrected codeword. This codeword is the one that minimizes the total cumulative distance between itself and the received sequence. Plots of SNR vs. P_e show a marked improvement in system performance, due to the post-processing algorithm. An analysis of the errors in the system provides evidence that the use of amplitude coding can still improve system performance.

CHAPTER 6

CODED MODULATION II: TRELLIS AND COMBINED RLL/ECC CODING

In Chapter 5, we observed that there is room for improvement in the performance of the recording system by adding an amplitude encoder. Until now, our approach for the amplitudes has been to use a simple symbol mapper. This mapper takes information bits and turns them into an M-ary symbol. However, we have not used error-correction. Therefore, an error in detection will cause multiple symbol errors. This effect can be mitigated if we use an error-correcting code on the amplitudes.

The use of an ECC on the amplitudes would allow us to correct simple error patterns. We assume that most amplitude error events are isolated. In general, they consist of one detected phrase for which its symbol value is in error. If this assumption holds, then the ECC will be able to correct the error patterns. This improves the overall performance of our encoding system.

6.1 Error Distribution Analysis

Tables 5 and 6, were used in our initial analysis in Chapter 5 to justify the use of an ECC code for amplitude encoding/decoding. Here, we will show that the possible improvements found in the $M = 5$ case are present in an even larger scale for the $M = 8$ case. That is, for $M = 8$ as the user bit density increases, so does the percentage of errors due to amplitude errors. In particular, this can be seen in Tables 7 and 8. Table 7 shows the analysis of errors for $M = 8$ and 1 bit/600nm. The SNR value on the Table is the value of SNR at the input the channel. It is not the SNR at the output, which is the one we use in our plots. To convert from one value to the other we need to add the value $\log_2 A_{h,n}^2$. Notice how at high SNR

Table 7: Error Distribution for $M = 8$, 1 bit/600nm.

SNR	P_e	Amp.Err	Symbols	Shift Err	Symbols	Total Errors
20.25	6.03322e-05	15	80	17	23	103
20.5	3.59321e-05	14	68	26	33	101
20.75	1.69636e-05	10	51	40	49	100
21	8.30618e-06	4	24	33	43	67
21.25	4.21508e-06	2	12	17	22	34
21.5	3.09932e-06	2	12	10	13	25

Table 8: Error Distribution for $M = 8$, 2.75 bit/600nm.

SNR	P_e	Amp.Err	Symbols	Shift Err	Symbols	Total Errors
20.25	8.0502e-05	12	75	23	30	105
20.5	5.25363e-05	14	82	15	22	104
21	1.76674e-05	10	54	38	49	103
21.25	7.99868e-06	6	27	20	26	53
21.5	3.55795e-06	4	19	19	22	41
21.75	1.90914e-06	2	12	8	10	22

values most of the errors in the system are due to symbol shift errors. These are errors that are not corrected by the Lee-Metric BCH code. Table 8 shows the error breakdown for $M = 8$ and 2.75 bits/600nm. For SNR at or below 21.75 dB, we observe that errors due to amplitude errors account for about fifty-percent of the total errors. These errors could be corrected if we use an ECC code on the amplitudes in our detection system.

6.2 Trellis Coded Modulation

Until now, our amplitude symbols have been obtained by a symbol mapper, mapping a set of bits into a set of M-ary amplitudes. We map the symbols individually if $\log_2(M-1)$ is an integer or in groups of N symbols so that $b_a = N\log_2(M-1)$ is an integer. There is no error correction in this approach. However, we have showed that introducing an error-correcting code can help improve system performance. There are some specific characteristics we have to keep in mind when selecting the code:

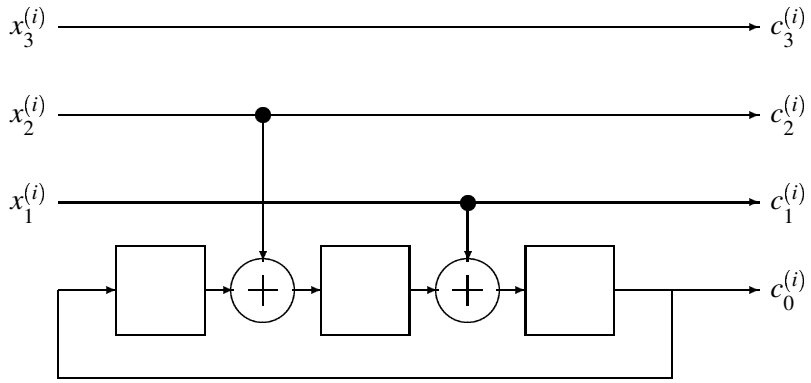


Figure 41: Systematic convolutional encoder with feedback for TCM

- The length of the codeword is short. In particular, for our system implementation we have only forty-eight phrases. Therefore, our codeword has to be at most forty-eight symbols long.

- The alphabet used for encoding has only $M-1$ symbols. It should not contain the symbol zero. If it does contain the zero symbol, translations of a codeword, where an integer number is added to each symbol should still be a codeword.

- Each codeword needs to be decoded individually. We require no look-ahead or look-back capabilities on the decoder.

We have decided to evaluate the performance of trellis coded modulation (TCM) for our simulations. TCM was first introduced in [48]. This reference provides a complete description of the encoding and decoding process.

We use this process to design a code for the $M = 5$ case. Our code then has $M - 1 = 4$ amplitude levels, with values in the following alphabet $\{1,\ 2,\ 34\}$. We use a systematic convolutional encoder with feedback to generate our code. The system diagram for our encoder is shown in Figure 41. We encode two amplitude symbols at a time and use three user bits to select these two symbols. Therefore, our encoding rate is one and a half bits/symbol. One uncoded bit is used to select the signal within the selected coset. The three coded bits

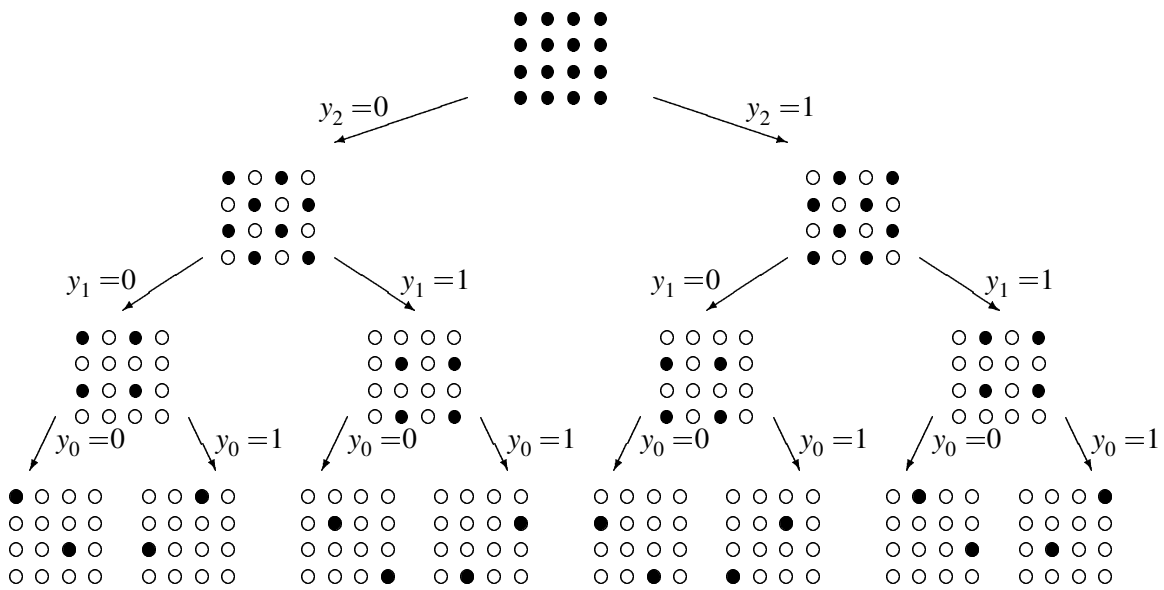


Figure 42: Coset partition for TCM alphabet

are used to select the coset in the alphabet. Figure 42 shows the alphabet used for our encoding process. It describes the process of coset partitioning and gives the final alphabet partition in our code. The symbol on the *x* axis represents the value of the first of the two amplitude symbols. The symbol on the *y* axis represents the value of the second of the two amplitude symbols. There are eight partitions in our alphabet, each consisting of two symbols.

The trellis for this code is an eight-state trellis diagram. Figure 43 shows the trellis for the implemented code. The numbers next to each trellis state represent the partition of the alphabet used in each of the transitions from that trellis state to another trellis state. The first number on the left is the partition selected on the lowest transition emanating from that trellis state. The last number on the right is the partition selected on the highest transition emanating from that particular trellis state. There are two parallel transitions per branch. These parallel transitions come from the uncoded bit used to select the specific symbol from the selected alphabet coset. Therefore, there are two pairs of alphabet symbols that can cause a transition from the current trellis state to one of the trellis states that can be

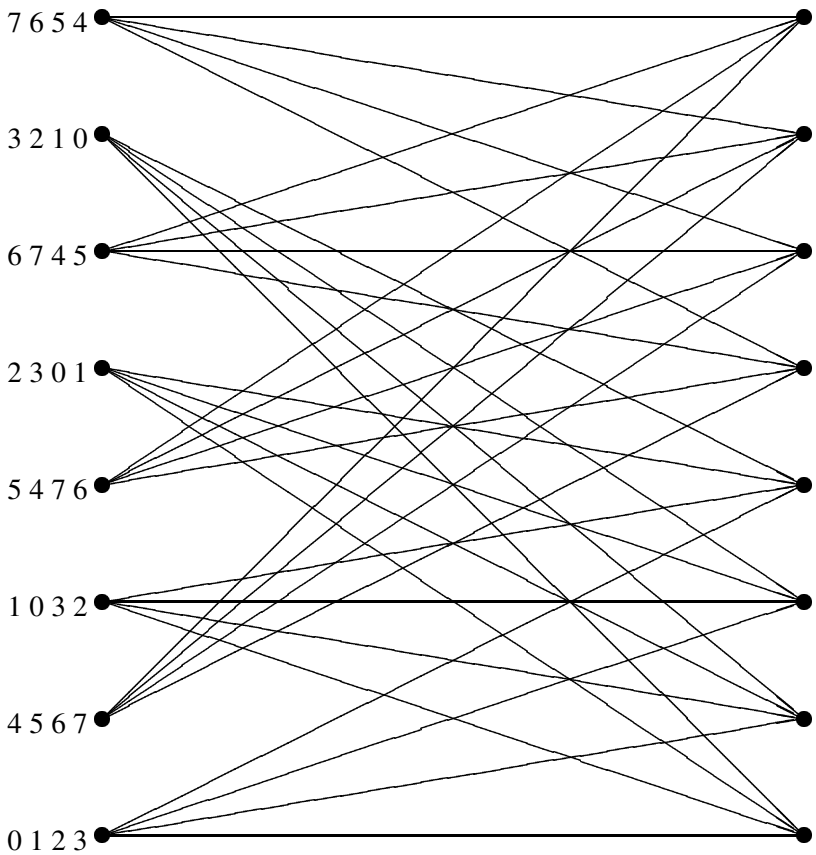


Figure 43: Trellis for TCM code.

reached from that trellis state in one trellis step.

6.2.1 Soft-Information for TCM Detection

The detector for decoding of the TCM coded amplitudes requires that soft-information be available for each amplitude symbol. This information is not directly available from the output of the Viterbi detector plus post-processing system. Therefore, we need to devise an approach to obtain the soft-information. We obtain this soft-information as follows:

- Use a zero-forcing linear-equalizer (ZF-LE) to filter the received signal and remove the ISI.
- From the detected codeword obtain the length of each phrase.

- For each phrase in the detected codeword add up the values of all received symbols in that phrase and divide by the total number of symbols in the phrase.

We then use the soft-information obtained in this manner as our received values for the trellis of the TCM detector.

6.2.2 The TCM Detector

The TCM detector is a modified Viterbi detector. The modification is necessary since the soft-information available to our detector is not matched to the alphabet of the TCM code, but to the channel alphabet. Therefore, to calculate the distance between the TCM symbols causing a transition in the trellis and the received soft-information we need to convert the TCM symbols into channel symbols.

For each trellis state we keep track of the last encoder state. The encoder state is the last symbol, y_i, output by the mod M precoder described in Section 2.3.1. The distance for a transition from one trellis state to another is calculated using the following steps:

- Select the partition that corresponds to the transition being evaluated.
- For each pair of symbols in the partition determine the corresponding channel symbols. This requires the last encoder state information saved for that trellis state. For example:
  - Assume we are at trellis state '0', $M = 5$, we are calculating the distance of the transition to trellis state '2' and the encoder state is '2'.
  - From Figure 43, the partition we use is $P = 2$. From Figure 42, the two pairs of symbols for this partition are $[2,2]$ and $[4,1]$.
  - For the first pair the mod M encoding gives $[4,1]$ and the last encoder state is '1'. For the second pair the encoding is $[1,2]$ and the last encoder state is '2'.
  - The pairs of channel symbols are $[4,-2]$ and $[-2,0]$ respectively.

Table 9: Error distribution for $M = 5$, 1 bit/600nm, without TCM detection.

SNR	P_e	Amp.Err	Symbols	Shift Err	Symbols	Total Errors
16	8.94889e-05	0	0	90	101	101
16.25	7.73318e-05	2	10	83	91	101
16.5	6.03696e-05	4	25	73	77	102
16.75	3.48693e-05	2	10	86	92	102
17	2.34838e-05	4	23	75	77	100
17.25	1.60835e-05	4	23	76	79	102
17.5	7.29353e-06	6	38	43	46	84
17.75	2.77849e-06	2	10	20	22	32
18	2.60483e-07	0	0	2	3	3

- Calculate the distance between each pair of channel symbols and the soft-information. Select the symbols closest to the soft-information and return the last encoder state corresponding to those symbols.

At each trellis state the normal compare and select procedure of Viterbi detection is used to guarantee we only keep the minimum distance path to that particular state. Therefore, the only modification from a normal Viterbi detector is the procedure to calculate the branch metric.

6.3 Results with TCM Amplitude Coding

Tables 9 and 10 show the new error data for $M = 5$ and 1 bit/600nm. We show the error data if no TCM error correction is performed in Table 9 and the data with the TCM error correction in Table 10. It can be seen that all amplitude errors are corrected when the TCM code is used. Therefore, our assumption that most amplitude errors are isolated and can be corrected with a TCM code is correct. Once again the simulations stop after one hundred symbol errors are made or forty thousand codewords are simulated. However, in the TCM case we stop after one hundred symbol errors without TCM correction. This means that we count, in the stopping criterion, the amplitude errors that our TCM code corrects.

Figure 44 shows the SNR vs. P_e for $M = 5$ at one and two bits/600nm. We compare the results with TCM coding of the amplitudes to the results from Chapter 5. We observe that

Table 10: Error distribution for $M = 5$, 1 bit/600nm, with TCM detection.

SNR	P_e	Amp.Err	Symbols	Shift Err	Symbols	Total Errors
16	8.94889e-05	0	0	90	101	101
16.25	6.96752e-05	0	0	83	91	91
16.5	4.55731e-05	0	0	73	77	77
16.75	3.14508e-05	0	0	86	92	92
17	1.80825e-05	0	0	75	77	77
17.25	1.24569e-05	0	0	76	79	79
17.5	3.99408e-06	0	0	43	46	46
17.75	1.91021e-06	0	0	20	22	22
18	2.60483e-07	0	0	2	3	3

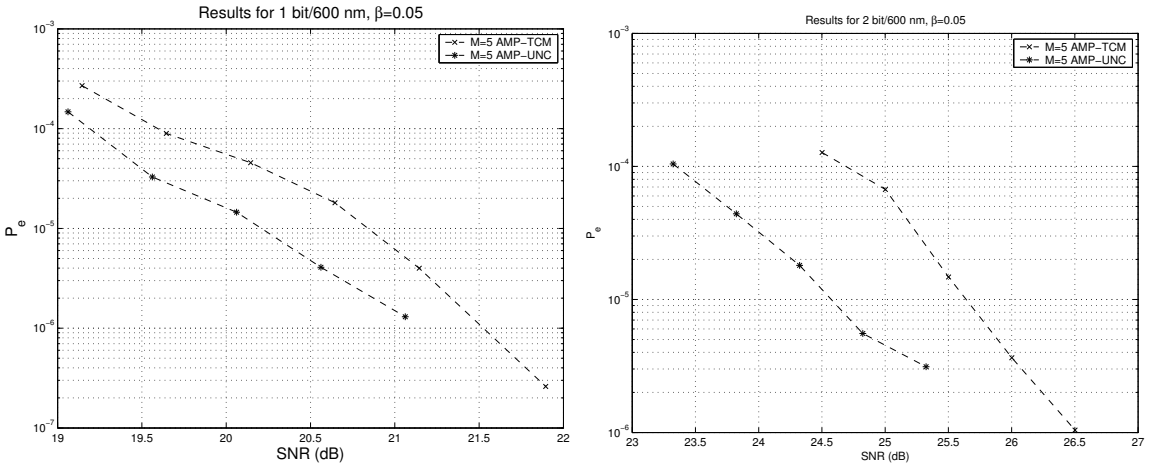


Figure 44: Results for TCM amplitude coded system.

the performance with the TCM code is worse than that without amplitude coding for both densities. There are two reasons for this behavior:

- With the TCM code we encode less bits per amplitude symbol than in the uncoded case. This reduction in rate means that we must signal with smaller marks to achieve the same user bit densities. The smaller marks mean we have more ISI in the channel. Therefore, even though we correct all amplitude errors that occur there is an increase in the number of symbol shift errors in the TCM coded signal.

- If all errors in our readout signal were amplitude errors, TCM coding would correct them and overcome the reduction in rate and increase in ISI. However, there are also

symbol shift errors in the channel. If the symbol shift errors are the main source of errors, then there are not sufficient amplitude errors for the TCM code to correct and it can not overcome the reduction in rate and increase in ISI and its performance is worse than that of the uncoded case.

We can see that the reduction in rate has a bigger effect when the user density is larger, in this case for the two bits/600nm case. This can be explained by the fact that the mark size for this system is smaller than that for the one bit/600nm system. Therefore, a further reduction in rate causes the marks to be even smaller, increasing the ISI and degrading performance.

There are two things we can learn from this analysis. First, TCM coding corrects the isolated amplitude errors encountered in the simulations. Second, the use of a TCM code is only justified if the percentage of total errors caused by amplitude errors is large. This occurs in general for lower SNR values and larger alphabet sizes ($M > 5$).

6.4 Increasing the Rate of the TCM Code

The TCM code corrects all the amplitude errors it encounters. However, it does not perform better than the uncoded system due to its loss of rate. For $M = 5$, the uncoded amplitudes encode two bits per amplitude symbol, while our TCM implementation encodes only one and a half bits per amplitude symbol. Therefore, it is necessary to increase the rate of the TCM code to improve performance. This is accomplished by encoding more symbols at a time. Here we describe an approach in which four symbols are encoded at a time. The size of our alphabet is now $4^4 = 256$. Therefore we use a rate $R = 4/5$ convolutional code to select one out of thirty two partitions. We then use three uncoded bits to select one of the eight alphabet symbols from the partition.

The convolutional code is described in [52]. It is a rate $R = 4/5$ code obtained by puncturing a rate $R = 1/2$ code. The rate one half code has a constraint length $\mu = 8$. The generator polynomials (in octal notation) of the code are given by $[561, 753]$. The

Table 11: Error distribution for $M = 5$, 1 bit/600nm, without TCM detection.

SNR	P_e	Amp.Err	Symbols	Shift Err	Symbols	Total Errors
15.5	0.00015468	0	0	77	84	84
16	6.6382e-05	0	0	74	82	82
16.5	3.3161e-05	0	0	61	74	74
17	1.07308e-05	0	0	54	63	63
17.5	3.90725e-06	0	0	38	45	45
18	1.5629e-06	0	0	15	18	18

puncturing masks are $[1101, 1010]$. This completely describes the code, with which we can encode 1.75 bits/amplitude symbol. We evaluate the performance of this code for the particular case

6.5 Results with Increased Rate TCM Amplitude Coding

Table 11 shows the error analysis for this higher rate TCM code. We observe that once again all amplitude errors are corrected by the TCM code. However, when comparing the number of symbol shift errors with those of the TCM code in Table 10, we observe that less errors are made with the new higher rate TCM code. Figure 45 shows the result of using the increased rate TCM code. In contrast with the results in Figure 44, we observe that for lower SNR values the performance of the TCM coded system is better than that of the amplitude uncoded system. Also at higher SNR values the TCM code system performance is less than one dB away from the uncoded system. The main reason for this improvement is the increase in the rate of our TCM code. This influences the results in two ways:

- A reduction in the rate loss of the TCM code directly influences the performance curves. There is less of a rate loss that the error correction capabilities of the code need to make up for.

- The increased rate means that we can signal with longer marks than in the previous TCM code case. This reduces the ISI present in the channel, which in turn reduces the number of symbol shift errors present in our simulations, as observed from Tables 10

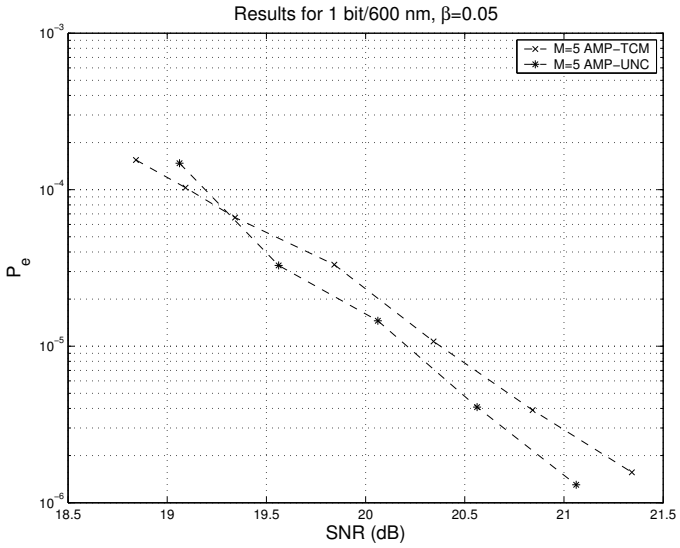


Figure 45: Results for increased rate TCM amplitude coded system.

and 11. Therefore, we obtain better performance with our TCM code.

This study shows that TCM coding in the amplitudes can outperform the uncoded amplitude system. However, for that to happen we require that the percentage of amplitude errors in the system be high, and that the TCM code be high rate. If the rate of the TCM code is not sufficiently high, then we can have performance degradation due to an increase in ISI, which in turns increases the number of symbol shift errors in the signal. This degradation is present even when all amplitude errors are corrected. If the percentage of amplitude errors in the system is not high enough, as is the case for high SNR values in Figure 45, then the error correcting code can not overcome the rate loss, even though it corrects all amplitude errors that are present. There are three parameters that affect the percentage of amplitude errors, they are:

- **SNR:** For lower SNR the percentage of amplitude errors is larger.
- ***M*:** For larger M the percentage of amplitude errors is larger.
- **User Density:** For larger densities (in bits/600nm), the percentage of amplitude errors is larger.

Therefore, we see that it makes sense to design TCM codes for systems with a large number of levels and a large user bit density and that will operate at low SNR values.

6.6 Conclusion

An error-correction code for the amplitude symbols was implemented. The ECC code is a trellis-coded modulation (TCM) code. It was implemented for the $M = 5$ system. It uses a systematic convolutional encoder with feedback, which generates a trellis with eight-states. Soft-information for the TCM code was generated using a ZF-LE and the phrase lengths from the detected BCH codeword. A modified Viterbi detector was used as the TCM decoder. Performance of the system was evaluated for user bit densities of one and two bits/600nm. The results show that TCM encoding of amplitudes provides a performance gain if the percentage of amplitude errors as a function of total errors is large. This occurs when we have a large number of levels M and a high user density and a low SNR value.

CHAPTER 7

CONCLUSION

7.1 Summary of Contributions

In this work, we proposed and evaluated a system for recording signals in an M-ary optical recording channel. The system uses a combined ECC/RLL code based on Lee-Metric BCH codes for binary RLL encoding. Amplitude Encoding is performed using trellis coded modulation (TCM).

Chapter 2 introduced the key technologies driving this research. It described the storage systems that allow or will allow the ideas in this research to be implemented in the near future. Some key concepts used in our system are also described.

In Chapter 3, we presented a continuous-time model for the optical recording channel. We calculated capacity for this model using the "waterfilling" method for capacity calculation. The model was then modified to account for the digital characteristics of the channel. Therefore, a discrete-time model was introduced. The parameter D which specifies the size of the recorded marks in the channel with respect to a minimum feature was introduced. Capacity plots were presented for this channel model and compared to those obtained with the continuous-time model. We also discussed the influence of other system parameters, such as α and β on the overall channel capacity.

Chapter 4 studied the performance of fixed-length and variable-length (RLL) recording on the discrete-time channel model. First, a sufficient-statistics discrete-time channel model was obtained. This model was designed so that it applies when both, fixed-length and variable-length marks, were recorded. The fixed-length mark systems included uncoded and TCM and Turbo coded systems. The variable-length recording system used high efficiency permutation codes. In the uncoded and permutation coded cases detection

was accomplished using a Viterbi detector. The trellis was based on the SSDT channel model. For the permutation coded case, the trellis incorporated the d constraint. It removed all states and transitions which violated the d constraint. It was observed that for user bit densities above 1 bit/600nm encoding information in the amplitude of the signal is recommended.

In Chapter 5, we introduced a complete encoding/decoding system using permutation codes and Viterbi detection. This system fails to decode the codeword when the Viterbi detector outputs a word that is not in the permutation code code space. Two possible solutions to this problem were investigated. First, joint maximum-likelihood (JML) detection of amplitudes and phrases was studied. It was found that this is not a viable solution due to the high complexity of the detection algorithm. We also investigated a post-processing algorithm for permutation codes. This process was also found to be too computationally intensive.

Section 5.3 presented an alternative solution to the problem encountered in Section 5.2. This solution involved changing the phrase encoder from a permutation encoder to a Lee-Metric BCH based encoder. The advantage of the Lee-Metric BCH encoder is an error-correcting and a runlength-limited encoder. Two different code construction methods were investigated. The algorithm introduced in [9], has the advantage that all codewords were of the same length. However, it suffers from a low encoding rate and the fact that no efficient encoding algorithm has been developed for this code. We decided to use the method introduced in [41]. The RLL codewords in this method have varying length. However, encoding and decoding are simple and efficient. Simulations were performed for the typical user bit densities. It was found that for low densities the algorithm performs well. However, for high user bit densities the performance degrades. This degradation is attributed to the Viterbi detection of words with the wrong number of phrases.

In Section 5.5, an algorithm for post-processing the output of the Viterbi detector from Section 5.3 is presented. This algorithm is introduced to correct detected words that have

the wrong number of phrases and/or phrases that exceed the k constraint. It has two components, one that corrects detected words with the wrong number of phrases and another that corrects phrases that exceed the k constraint. The task of correcting for the wrong number of phrases is split into two separate blocks. One block adds phrases to a codeword and the other removes phrases from a codeword. The goal of the post-processing algorithm is to find a word, with the correct number of phrases, that minimizes the total cumulative distance between itself and the received sequence. The results using this algorithm show a dramatic improvement over the results in Section 5.4. Once again we observe that M-ary encoding outperforms binary encoding for user bit densities above 1 bit/600nm. The results also show that the overall system performance can be improved with the use of amplitude encoding.

Finally, Chapter 6 introduced a method for amplitude encoding on the recorded signal. The method is based on trellis coded modulation (TCM). Its performance is presented for the special case of $M = 5$. This because obtaining a code for this amplitude value is simple and its implementation is also much faster than if another M value is used. The results show that TCM encoding of the amplitudes provides an advantage over the system with uncoded amplitudes if the percentage of amplitude errors as a function of total errors is large. Therefore, one area of future research should be the implementation of this technique for other M values.

7.2 Directions for Future Research

There is still considerable work to accomplish in this research field. This work in no way attempts to solve all the problems in this area or provide any definitive solution. On the contrary, it attempts to be a stepping stone for future research in the area. We attempt to provide a guiding light as to what are some good ideas to investigate and what roads not to take. For our research in particular, we feel that a more thorough evaluation of TCM coding on the amplitudes needs to be performed. We have considered only the $M = 5$ case in our

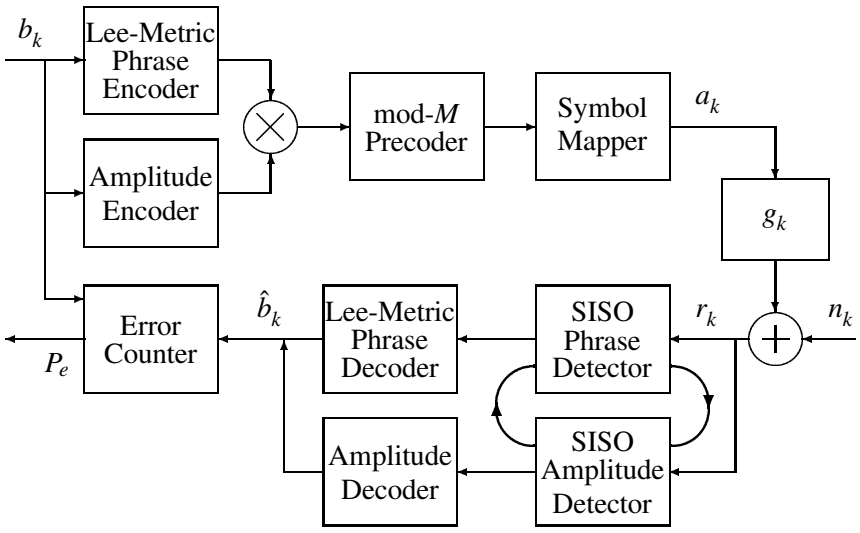


Figure 46: Permutation Code System with Iterative Decoding

evaluation. We feel that a complete evaluation would design TCM codes for both $M = 6$ and $M = 8$. These values of M are the ones that seem to benefit the most from amplitude encoding.

In our research, we selected Lee-Metric BCH codes as our combined ECC/RLL codes for their simplicity and their ability to combine error-correction and runlength-limited encoding into one code. However, this is not the only way that the ECC-RLL combination can be approached. In particular, reverse concatenation (RLL-ECC) has been shown to work efficiently in binary systems. Therefore, another area of research is the study of reverse concatenation for the M-ary optical data storage channel. This method might achieve performance gains that the combined ECC/RLL codes might not provide.

Finally, Figure 46 shows the block diagram for a novel decoding system. This system performs iterative detection before the code decoders. We believe that the availability of soft-information from the detectors and the capability to iterate between the phrase and amplitude detectors, can provide performance gains. Therefore, it is of interest to evaluate the performance of such a system.

The two SISO detectors in this system should be able to communicate between them,

providing soft-information that the detector can use as a prior log-likelihood in its iteration. The detection process would now be similar to the iterative decoding used in Turbo codes. The implementation of the SISO amplitude detector is simple. Such algorithms have been used extensively in the literature. In particular, a combination of BCJR with a soft demapper for multilevel symbols was described in [45]. This could be a starting point for a specific implementation. Another approach, described in the same reference, is to include multilevel symbols into the distance metric of a BCJR decoder. This decoder is then based on channel symbol likelihood ratios rather than bit symbol likelihood ratios.

A SISO detector for the phrases is more difficult to implement. In particular, soft-information on the length of a particular phrase needs to be obtained. With our channel model, this type of timing information is unavailable. Therefore, additional signal processing of the received signal would be required to provide soft-information on the phrase lengths. If this information is available, then a SISO phrase detector can be implemented. The length of a phrase then provides soft-information on the symbol amplitudes, since we know that the amplitude must be constant for the duration of the phrase. Therefore, the communication between the two SISO detectors should be simple to perform, if both detectors exist.

REFERENCES

[1] ANIM-APPIAH, K. D. and MCLAUGHLIN, S. W., "Towards soft output app decoding for nonsystematic nonlinear block codes," in *Proceedings 1999 Allerton Conference on Communications Control and Computing*, (Urbana, IL), pp. 1304–1313, 1999.

[2] BAERT, L., THEUNISSEN, L., and VERGULT, G., eds., *Digital Audio and Compact Disc Technology*. Oxford: Butterworth-Heinemann Ltd, 2nd ed., 1992.

[3] BAHL, L., COCKE, J., JELINEK, F., and RAVIV, J., "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. on Information Theory*, vol. 20, pp. 284–287, Mar. 1974.

[4] BARBOSA, L., "Minimum noise pulse slimmer," *IEEE Trans. on Magnetics*, vol. 17, pp. 3340–3342, Nov. 1981.

[5] BERGMANS, J. W. M., *Digital Baseband Transmission and Recording*. Dordrecht, The Netherlands: Kluwer, 1996.

[6] BERLEKAMP, E. R., *Algebraic Coding Theory*. Laguna Hills, CA: Aegean Park Press, revised ed., 1984.

[7] BLAUM, M., "Combining ECC with modulation: performance comparisons," *IEEE Trans. on Information Theory*, vol. 37, pp. 945–949, May 1991.

[8] BORG, H. and VAN WOUDENBERG, R., "Trends in optical recording," *Journal of Magnetism and Magnetic Materials*, vol. 193, pp. 519–525, 1999.

[9] BOURS, P. A. H., "Construction of fixed-length insertion/deletion correcting runlength-limited codes," *IEEE Trans. on Information Theory*, vol. 40, pp. 1841–56, Nov. 1994.

[10] BRANDENBURG, L. and WYNER, A., "Capacity of the gaussian channel with memory: the multivariate case," *Bell System Tech. J.*, vol. 53, pp. 745–778, may-jun 1974.

[11] COENE, W., POZIDIS, H., and BERGMANS, J. W. M., "Run-length limited parity-check coding for transition-shift errors in optical recording," in *IEEE Globecom*, vol. 2, (San Antonio, TX), pp. 2982–86, Institute of Electrical and Electronics Engineers, 2001.

[12] COENE, W., POZIDIS, H., VAN DIJK, M., KAHLMAN, J., VAN WOUDENBERG, R., and STEK, B., "Channel coding and signal processing for optical recording systems beyond DVD," *IEEE Trans. on Magnetics*, vol. 37, pp. 682–688, mar 2001.

[13] COVER, T. M. and THOMAS, J. A., *Elements of Information Theory*. New York: John Wiley & Sons, 1991.

[14] DATTA, S. and MCLAUGHLIN, S. W., "Optimal block codes for m-ary runlength-constrained channels," *IEEE Trans. on Information Theory*, vol. 47, pp. 2069–78, July 2001.

[15] DATTA, S. and MCLAUGHLIN, S., "An enumerative method for runlength-limited codes: permutation codes," *IEEE Trans. on Information Theory*, vol. 45, pp. 2199–2204, Sept. 1999.

[16] EARMAN, A., "Optical data storage with electron trapping materials using M-ary data channel coding," in *Optical Data Storage*, vol. 1663, (San Jose, CA, USA), pp. 92–103, 1992.

[17] FERREIRA, H. C. and LIN, S., "Error and erasure control (d,k) block codes," *IEEE Trans. on Information Theory*, vol. 37, pp. 945–49, May 1991.

[18] FRENCH, C., DIXON, G., and WOLF, J., "Results involving (D,K) constrained M-ary codes," *IEEE Trans. on Magnetics*, vol. 23, pp. 3678–3680, Sept. 1987.

[19] GALLAGER, R., *Information Theory and Reliable Communication*. New York: John Wiley & Sons, 1968.

[20] GOLDSMITH, P., LINDMAYER, J., and WRIGLEY, C., "Electron trapping, A new approach to rewritable optical data storage," in *Proc. of SPIE, Optical Data Storage*, vol. 1316, (Vancouver, BC, Canada), pp. 312–320, 1990.

[21] HEEMSKERK, J. and IMMINK, K., "Compact disc: system aspects and modulation," *Philips Journal of Research*, vol. 40, no. 6, pp. 157–164, 1982.

[22] HILDEN, H., HOWE, D., and JR., E. W., "Shift error correcting modulation codes," *IEEE Trans. on Magnetics*, vol. 27, pp. 4600–4605, Nov. 1991.

[23] HOLLMANN, H., "On the construction of bounded-delay encodable codes for constrained systems," *IEEE Trans. on Information Theory*, vol. 41, pp. 1354–1378, Sept. 1995.

[24] IMMINK, K., SIEGEL, P., and WOLF, J., "Codes for digital recorders," *IEEE Trans. on Information Theory*, vol. 44, pp. 2260–2299, Oct. 1998.

[25] KATZ, E. and CAMPBELL, T., "Effect of bitshift distribution on error rate in magnetic recording," *IEEE Trans. on Magnetics*, vol. 15, pp. 1050–53, May 1979.

[26] KAUTZ, W., "Fibonacci codes for synchronization control," *IEEE Trans. on Information Theory*, vol. 11, pp. 284–292, Apr. 1965.

[27] LEE, C. Y., "Some properties of nonbinary error-correcting codes," *IEEE Trans. on Information Theory*, vol. 4, pp. 77–82, June 1958.

[28] LEE, E. and MESSERSCHMITT, D., *Digital Communication*. Boston: Kluwer, 1998.

[29] MARCHANT, A. B., *Optical Recording: A Technical Overview*. New York: Addison-Wesley, 1990.

[30] MARX, D. and PSALTIS, D., "Pit depth encoded memories," in *Proc. of SPIE, Optical Data Storage*, vol. 2338, (Dana Point, CA, USA), pp. 77–78, 1994.

[31] MCLAUGHLIN, S. W., LEE, D., LO, T., PEPIN, C., SUPNITHI, P., and WARLAND, D., "Advanced coding and signal processing for multilevel write-once and rewritable optical storage," in *Optical Data Storage Topical Meeting*, (Santa Fe, New Mexico), Optical Society of America, 2001.

[32] MCLAUGHLIN, S. W., LO, Y., PEPIN, C., and WARLAND, D., "Multilevel DVD: Coding beyond 3 bits/data-cell," in *International Symposium on Optical Memory and Optical Data Storage Meeting*, (Waikoloa, Hawaii), pp. 380–382, Optical Society of America, 2002.

[33] MCLAUGHLIN, S., "Improved distance M-ary (*d,k*) codes for high density recording," *IEEE Trans. on Magnetics*, vol. 31, pp. 1155–1160, Mar. 1995.

[34] MCLAUGHLIN, S., "Five runlength-limited codes for M-ary recording channels," *IEEE Trans. on Magnetics*, vol. 33, pp. 2442–2450, May 1997.

[35] MINAGAWA, N. and HAYASHI, H., "Multilevel signal processing technique for a 30Gbit/inch^2 phase change optical disk system," in *International Symposium on Optical Memory (ISOM)*, (Waikoloa, Hawaii), pp. 278–280, Optical Society of America, 2002.

[36] O'NEILL, M., BALASUBRAMANIAN, K., WEBER, T., HORIE, M., KIYONO, K., and MIZUNO, M., "Over 20GB rewritable multi-level recording using blue laser and growth-dominant phase-change optical discs," in *International Symposium on Optical Memory (ISOM)*, (Citose, Japan), Optical Society of America, 2000.

[37] O'NEILL, M. and WONG, T., "Multi-level data storage system using phase-change optical discs," in *Optical Data Storage Topical Meeting*, (Whistler, BC, Canada), pp. 170–2, Optical Society of America, 2000.

[38] PÁTROVICS, L. and IMMINK, K., "Encoding of *dklr*-sequences using one weight set," *IEEE Trans. on Information Theory*, vol. 42, pp. 1553–1554, Sept. 1996.

[39] PLESS, V. and HUFFMAN, W., *Handbook of Coding Theory*, vol. 2. Amsterdam, The Netherlands: Elsevier, 1998.

[40] PROAKIS, J. and TYNER, D., "Partial response equalizer performance in digital magnetic recording channels," *IEEE Trans. on Magnetics*, vol. 29, pp. 4192–4208, Nov. 1993.

[41] ROTH, R. M. and SIEGEL, P. H., "Lee-metric BCH codes ant their application to constrained and partial-response channels," *IEEE Trans. on Information Theory*, vol. 40, pp. 1083–96, July 1994.

[42] SHIMAZAKI, K., YOSHIHIRO, M., ISHIZAKI, O., and OHTA, N., "Direct overwriting capability of magnetic multi-valued mo media," in *Proc. of SPIE, Optical Data Storage*, vol. 2514, (San Diego, CA, USA), pp. 62–63, 1995.

[43] TAKESHITA, O., KOHNO, R., and IMAI, H., "Multilevel RLL (*D,K,l*) constrained sequences," *IEICE Trans. Fund. of Electr., Comm. and Comp. Science*, vol. E-77A, pp. 1238–1245, Aug. 1994.

[44] TANG, D. and BAHL, L., "Block codes for a class of constrained noiseless channels," *Information and Control*, vol. 17, pp. 436–461, 1970.

[45] TEN BRINK, S., SPEIDEL, J., and YAN, R.-H., "Iterative demapping and decoding for multilevel modulation," in *IEEE Globecom*, vol. 1, (Sydney, Australia), pp. 579–84, Institute of Electrical and Electronics Engineers, 1998.

[46] TSYBAKOV, B., "Capacity of a discrete time gaussian channel with a filter," *Problem y Peredachi Informatsii*, vol. 6, pp. 78–82, jul-sep 1970.

[47] ULRICH, W., "Non-binary error correction codes," *Bell System Tech. J.*, vol. 36, pp. 1341–1387, Nov. 1957.

[48] UNGERBOECK, G., "Channel coding with multilevel/phase signals," *IEEE Trans. on Information Theory*, vol. 28, pp. 55–67, Jan. 1981.

[49] VITERBI, A., "Error bounds for convolutional codes and asymptotically optimum decoding algorithm," *IEEE Trans. on Information Theory*, vol. 13, pp. 260–269, Apr. 1967.

[50] WILLIAMS, E. W., *The CD-ROM and Optical Disc Recording Systems*. New York: Oxford University Press, 1994.

[51] WU, K., HOWE, D., and TSAI, S.-Y., "Recording of multi-level run-length-limited (ML-RLL) modulation signals on phase-change optical discs," in *Optical Data Storage Topical Meeting*, (Vancouver, BC, Canada), Optical Society of America, 2003.

[52] YASUDA, Y., KASHIKI, K., and HIRATA, Y., "High-rate punctured convolutional codes for soft decision Viterbi decoding," *IEEE Trans. on Communications*, vol. 32, pp. 315–319, Mar. 1984.

[53] YTREHUS, O., "Runlength-limited codes for mixed-error channels," *IEEE Trans. on Information Theory*, vol. 37, pp. 1577–1585, Nov. 1991.

[54] ZHOU, T., TAN, C., LEIS, C., HARVEY, I., LEWIS, G., WONG, T., and O'NEILL, M. P., "Multlevel amplitude-modulation system for optical data storage," *Optical Data Storage Topical Meeting*, pp. 170–2, 2000.

VITA

Estuardo Licona was born in Guatemala city, Guatemala on December 25, 1973. He received his B.S. and Licenciado degrees in Electronic Engineering from the Universidad del Valle de Guatemala, Guatemala, in 1995 and 1998, respectively. In August of 1997 he received the Masters of Engineering degree in Electrical Engineering from Cornell University, Ithaca, New York. In September 1997, he started attending the School of Electrical and Computer Engineering at the Georgia Institute of Technology in Atlanta, Georgia. He obtained his M.S. degree in 1999 and the Ph.D. degree in 2004. During his time at Georgia Tech he held positions as research and teaching assistant. His current research interests include the applications of error control coding and signal processing to communication and medical systems.